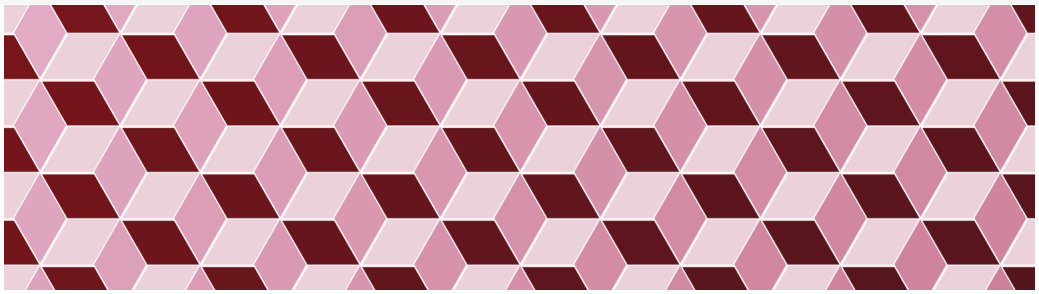

15-104 Introduction to Computing for Creative Practice

Fall 2021

28 Program Development

Instructor: Tom Cortina, tcortina@cs.cmu.edu, GHC 4117, 412-268-3514

1

Specifications

- Every program should have two parts:
  - A *specification*, usually informal text in comments, that explains how and why your program works.
  - The *implementation*, i.e. the code.
- Rather than making the code solve your problem directly, you should focus on making the code do exactly what your *specification* says it does.
  - As you work, you will find both errors in the specification *and* in the implementation. The problem with writing code directly with English descriptions is that code is too detailed and hard to reason about. Having two perspectives on the problem and its solution is invaluable.

2

Example: Turtle Graphics

- Specification: API documentation:

```
// makeTurtle(x, y) -- make a turtle at x,y, facing right, ...
// left(d) -- turn left by d degrees
// right(d) -- turn right by d degrees
// forward(p) -- move forward by p pixels
// back(p) -- move back by p pixels                    etc.
```

- Implementation:

```
function makeTurtle(tx, ty) {
    var turtle = {x: tx, y: ty,
                  angle: 0.0,
                  penDown: true,
                  color: color(128),
                  weight: 1,                          etc.
```

3

Incremental Development

- Start with a simple program that works and modify it incrementally, adding new features one-at-a-time until your program is finished.
- Test at every step.
  - If you encounter a problem, it will almost always be related to the last change you made, which is both small and fresh in your mind, and therefore quick and easy to fix.
- Save/archive your work regularly in case you need to back up to a previously working version. (Programmers use version control systems like **git** for this: https://git-scm.com/)

4

Rubine's Rule

- If you find a bug, fix it. (Due to Dean Rubine, programmer extraordinaire, PhD CMU.)
- Programmers often think they can pursue one problem while ignoring other "unrelated" problems.
  - Those "unrelated" problems are often critical and your misunderstanding of them will cause all sorts of confusion.
  - To avoid getting nailed by your own misunderstanding: if you find a bug, fix it.

5

Write Quality Code

- Do not tolerate sloppy code, especially if it is yours.
- In the words of a former student, you should love yourself and therefore love your code, which is yet another manifestation of who you are.
- In a way, good code, good art, and good writing are all part of being the person you want to be.
- Making things of quality may seem more difficult, but if you develop a strong habit of being careful, demanding, critical and exacting, you will find that it actually makes coding easier.

6

Using Multiple Source Files

- Larger programs are usually written using multiple files.
- Even p5.js complete libraries come in three files (which is why we have "min" and "all" templates).
- Our current index.html file:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>p5.js vers 0.5.12, Edit index.html to Change This Title</title>
    <script src="http://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.12/p5.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.12/addons/p5.dom.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.12/addons/p5.sound.js"></script>
    <script src="sketch.js" type="text/javascript"></script>
  </head>
  <body>
  </body>
</html>
```

7

Using Multiple Source Files (cont'd)

- To include multiple source files, you only need to insert additional script lines in your index.html.
  - The following does *not* use sketch.js, but instead runs two files — mybigproject.js and drawingfunctions.js:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>p5.js vers 0.5.12, Edit index.html to Change This Title</title>
    <script src="http://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.12/p5.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.12/addons/p5.dom.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.12/addons/p5.sound.js"></script>
    <script src="mybigproject.js" type="text/javascript"></script>
    <script src="drawingfunctions.js" type="text/javascript"></script>
  </head>
  <body>
  </body>
</html>
```

The resulting program will consist of all the top-level (global) function and variable definitions. The preload, setup, and draw functions will be called as usual, regardless of where they are defined.

8

Example (see code sample zip file)

- Store the code for the Turtle in a separate file: turtle.js.
- Store the setup and draw function in sketch.js, along with any other functions you write, if any.
- To load both files along with the p5.js library to form the program to run, edit the index.html file to load all three components (the order doesn't matter):

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>p5.js vers 0.9.0, Edit index.html to Change This Title</title>
    <script src="http://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/p5.js"></script>
    <script src="turtle.js" type="text/javascript"></script>
    <script src="sketch.js" type="text/javascript"></script>
  </head>
  <body>
  </body>
</html>
```

9

Final Project

- Goal: Using the programming skill you have gained throughout the semester, you are to create a project of your own design based on the theme of Climate Change.
- Your project should be a program that expresses an artistic interpretation of this theme that is designed to start a discussion.
- Although you are free to express your views through your (programmed) art, you should be aware that you will be posting this publicly. Please avoid obscenities, nudity, blasphemy, slander, etc.

10

Final Project

- As you think about what you want to do, draw sketches and take notes of ideas you have. You should generate 3-5 ideas that you can then whittle down to your desired project.
- Think about projects that you can reasonably do given the programming knowledge you have gained.
  Don't bite off a lot more than you can chew.
- Your program should illustrate correct understanding of the following concepts: objects, loops, arrays, conditionals (if), transformations, functions (besides setup and draw).
- **Work incrementally, test carefully and give yourself enough time.**

11

Final Project

- **Proposal will be due Saturday, Nov. 20 as part of Deliverable Week 12.**
- We will review your proposals during the Thanksgiving week and provide feedback by Tuesday, November 23.
- **The final project will be due on FRIDAY, DECEMBER 3 by 11:59PM. (You will need to submit a checkpoint on Saturday, Nov. 27.)**
- All projects will be submitted to Autolab for grading and to the WordPress site for public view.
- Collaboration: Students may work independently or in teams of 2.
  - For proposals with 2 students, the proposal MUST indicate the individual contributions of each student.

12

Plagiarism

- **Your work must be an original idea and creation.**

- If you are inspired by some computational art, please cite this in your proposal.
- It is ok to be informed by an artist's work, but it is not ok for this project to duplicate that work (or use it as your own).
- We will be checking resulting projects for similarity with work available online. Students who clearly plagiarize work will be charged with an academic integrity violation and will fail this class.
- The work is also expected to be your own coding. Students who copy someone else's code as their own or have someone code for them can be charged with an academic integrity violation and will fail this class.

13