Odds and Ends

• Please submit any images files you used along with your program
• Name your screenshot something very obvious – like “screenshot.jpg”
• Do not leave any files scattered in your Dropbox folder. It needs to be in an assignment folder or I won’t know which assignment it belongs to!
• Name all your assignment folders well, like assignment01, sketch01, etc

Review

• Variable declarations
• Variable assignments
• Loops
  – Condition
  – index
• Functions
  – Definition
  – Call
  – Parameters

Execution

• Statements are executed one at a time in the order written

• Execution order
  – Global variables and initializations
  – setup() called once
  – draw() called repeatedly (unless noLoop() is called in setup())
  – If any mouse or keyboard events occur, the corresponding functions are called between calls to draw() – exact timing cannot be guaranteed.

Identify Similar Code

```
void drawRandomRect() {
  fill(random(255), random(255), random(255), 50);
  x = random(width);
  y = random(height);
  w = random(5, 100);
  h = random(5, 100);
  rect(x, y, w, h);
}

void drawRandomCircle() {
  fill(random(255), 50);
  x = random(width);
  y = random(height);
  w = random(5, 100);
  h = random(5, 100);
  ellipse(x, y, w, h);
}
```

Functions that return values

• The return value of a function is the output of a function.
• A function evaluates to its return value.
• Function must return a value whose type matches the function declaration.

```
return_type function_name(parameter_list) {
  statements;
  return value;
}
```
Example

What is the value of `result` in each line?

```cpp
void setup () {
  result = A(2);
  result = B(1, 2);
  result = 10 + A(2);
  result = A(2) + B(1, 2);
  result = B(A(2), B(B(1, 2), A(2)));
}

int A(int x) {
  return x*2;
}

int B(int x, int y) {
  return x+y;
}
```

Variable Lifetime

- Variables cannot be referenced before they are declared.
- A variable is created and initialized when a program enters the block in which it is declared.
  - Functions
  - Loops
  - Conditionals
  - Function parameters
- A variable is destroyed when a program exits the block in which it was declared.

Variable Scope

- The region of code in which a particular variable is accessible.
- To a first approximation, the scope of a section of your code is demarcated by `{` and `}`.
  - Functions
  - Loops
  - Conditionals
- A variable is only accessible/available within the scope in which it is declared.

Global variables

- Variables that are declared outside of any scope are considered globals (versus locals).
- Global variables should be declared at the top of your program.
- Do not sprinkle them between functions!

Shadowing

- When there is a name conflict between variables of different scopes
  ```cpp
  int x = 10;
  void setup() {
    int x = 5;
    int y = x;
  }
  ```
- The conflicting variables can not have different types (or it’s considered a re-declaration and is not allowed)
- When shadowed, smaller (inner) scopes have precedence over larger (outer) scopes

```cpp
int v1 = 1;
void setup() {
  int v0 = 2;
  for (int v3 = 0; v3 < 3; v3++) {
    int v4 = 4;
    println("------setup------");
    println(v1);
    println(v0);
    println(v3);
    println(v4);
    println(v4);
  }
  int v3 = 6;
  println(v3);
  println(v3);
  println(v3);
  println(v3);
  println(v3);
  println(v3);
  println(v3);
  println(v3);
}
void aFunction(int v5) {
  println("------aFunction------");
  println(v1);
  println(v1);
  println(v1);
  println(v1);
  println(v1);
  println(v1);
  println(v1);
  println(v1);
}
```
Example
- scoping

Code tracing
- We learn to read code by executing the code line by line
- Do not jump ahead
- Do exactly what the code says, step by step
- Keep a diagram of all variables and update them accordingly
- Mistakes are almost always due to skipping steps

Trace this
```
1 int n = 365;
2 int sum = 0;
3 int digit;
4 while(n>0) {
5    digit = n%10;
6    sum += digit;
7    n /= 10;
8 }
9 println(sum);
```

Nested loops
- You can put a loop within a loop
- Nesting levels are unlimited, but in practice programmers rarely go beyond 3
- Two loops nested is very common, especially when dealing with naturally 2-dimensional structures (grids)
  - for(...){
    - for(...){
      - while(...){
        - while(...){
          - for(...){
            - for(...){
```

Nested for
```
int i, j, end = 10;
for (i = 1; i <= end; i++) {
    for (j = i; j <= end; j++) {
        print("*");
    }
    println();
}
```

Nested for
```
int i, j, end = 10;
for (i = 1; i <= end; i++) {
    for (j = 1; j <= i; j++) {
        print("*");
    }
    println();
}
Examples

• pictureTile
• pictureTile2
• gradientWhileLoop