Ex. Color-Loop 3

Ex. Shade

Exercises: re-write all nested loops as

- while within while
- while ... to
- for ... while
do-while loop:

\[\text{do}\)
\[\text{if}\]
\[\text{start};\]
\[\vdots\]
\[\text{start};\]
\[\text{while} (\text{cond});\]
\[\text{start};\]
\[\vdots\]
\[\text{end}\]

**Note:** Loop body must execute at least one time.

Ex: Simple Loop 2
Exercise: re-write all previous examples as do-while.

Chapter 7: Functional

Functions are a means of breaking up a complex task into smaller, smaller, smaller pieces.

Also: a function is a named set of instructions.
Two contexts for functional:

- Function definitions.
- Function calls.

```c
#define
/ return type. list of formal parameters

__type __funcName typ1 var1, typ2 var2, ... {
  // Starts to execute when func is called.

  return exp; // absent if
  // return type
  // is void
}
```
Ex. Function 1

Ex. Function 2

```
function Call:   argument list

funcName (exp1, exp2, ...);
(If return type void)
```

```
var = funcName (exp1, exp2, ...);
(If return type non-void)
```
when a fn is called

- Slot is allocated for formal parameters \( \text{var1, var2} \). . .

- Values of expressions in argument list are copied to formal parameters

- Execution transfers to body at fn definition.

- If fn is non-void return type, then a return stmt. must be reached. The value of the \( \text{exp} \) is the value returned by the fn.

- Memory for fn params. is de-allocated.
execution transfers back to calling function.

we've been calling lots of fns, built into necessary

- line()
- ellipse()
- rect()

we've also been defining some

built in fns.

- setup()
- draw()
- mousePressed()
Ex. Functions 3