Midterm Review, Function, For loops

Announcements

- Creative Assignment HW9
  Deadline extended
  Tuesday, Nov. 13th before class

- Will cover some stuff today and next Tuesday that
  might help you be more creative

- Midterm Thursday. Will cover all the material prior
to today.

HW7: Creativity in Processing

Assignments: Write four programs to do whatever you want (but don’t copy the examples above), and try to make them clever or interesting or cute or have some property that would interest a viewer. You should try to use those you have learned in your former Processing homework, because those are the basics and one goal of this assignment is to practice the basics. But, if you need some other feature of processing that you find in the reference page, go ahead and use it. The goal is creativity ... but don’t spend forever on it either.

Midterm Review, Function, For loops

What will be on the midterm?

Programming

- LightBot
- Processing
  - Be able to read a Processing sketch and figure out what it
does.
  - draw(), setup(), shape drawing functions, variables, if-else

Bits and Bytes

- Binary numbers
- Logic gates
- Transistors
- Representing data: pictures, text, numbers,
documents
- Huffman Trees
- Cloud computing
- Meta data
- Steganography
- !!!
Other
- Seven Big Ideas in Computing
- Blown to Bits – Ghosts in the machine
- LifeLogging
- Privacy in a digital world

The Seven Big Ideas in computing
- As defined by the College Board for new AP test
  1. Computing is a creative human activity that enables innovation
  2. Abstraction is a way to understand and solve problems
  3. Data and information help to create knowledge
  4. Algorithms are tools for developing and expressing solutions to computational problems
  5. Programming is a creative process that produces computational artifacts
  6. Digital devices, systems, and the networks that interconnect them enable and foster computational approaches to solving problems
  7. Computing enables innovation in other fields, like science, engineering, humanities, etc.

Instructions Formed of Simpler Instructions
- Check out this screen shot of the Lightbot
- It is partway through an instruction … its beacon is lit, but not the tile
- To a programmer the instruction is monolithic (one thing)
- To an agent each instruction is a series of steps

Abstraction
- The word “abstraction” is used a lot in computing
- Remember: it was one of the 7 big ideas
- Abstraction is a way to understand and solve problems
- As a general definition, abstraction eliminates details to focus on essential properties
- The instruction example just given illustrates functional abstraction meaning that we have given a name to a series of operations that perform a coherent (and to us meaningful) activity; the name is the instruction, the series of operations are the bot’s actions to implement it

Abstracting
- Collecting the operations together and giving them a name is functional abstraction
- The group of operations perform some function but we ignore all of the details
- Giving it a name is functional abstraction
- This is AMAZINGLY powerful
- What makes it powerful, is we can forget about the operations and think only about the function they do, more about this later
- Let’s do some functional abstraction
Functions, A Review

- Functions have been used in Lightbot 2.0: F1
- Functions were in Assignment 02: F.turn()
- We’ve used functions, also known as
  - procedures
  - methods
  - subroutines

In all of our Processing code: size(200, 200)

Recall that functions have two parts:
- function definition — a statement of how it works
- function call — a request to have it performed

Calling vs Defining

```java
void draw() {
  background(255);
  rect(20,20,50,60);
}
```

Which best describes this program?
A. It defines one method and calls two.
B. It calls one method and defines two.
C. It defines three methods.

Calling vs Defining

```java
void drawTest() {
  background(255);
  rect(20,20,50,60);
}
```

Which best describes output of this program?
A. It displays the outline of a rectangle on a white, 100x100 background.
B. It displays a default grey square (100x100).

```java
void draw() {
  background(255);
  rect(20,20,50,60);
}
```

```java
void drawTest() {
  background(255);
  rect(20,20,50,60);
}
```
There must be a better way

- What if I decide to modify the look of the car a bit? I have to make any changes in 3 places.
- What if I want yet another car? I have to copy and paste a big chunk of code.
- You have to scan a moderate amount of code to determine I’m drawing 3 cars.

```java
void setup() {
  size(600, 400);
  drawCar(100, 100, 200);
  drawCar(300, 300, 150);
  drawCar(350, 100, 100);
}

void drawCar(int frontOfCar, int topOfCar, int lengthOfCar) {
  int carBodyHeight = lengthOfCar / 4;
  int wheelDiameter = lengthOfCar / 4;
  // draw the body
  rect(frontOfCar, topOfCar, lengthOfCar, carBodyHeight);
  // draw the wheels
  ellipse(frontOfCar + wheelDiameter, topOfCar + carBodyHeight, wheelDiameter, wheelDiameter);
  ellipse(frontOfCar + lengthOfCar - wheelDiameter, topOfCar + carBodyHeight, wheelDiameter, wheelDiameter);
  // draw the windshield
  // use carBodyHeight to also control the size of the windshield
  line(frontOfCar + carBodyHeight, topOfCar, frontOfCar + 2 * carBodyHeight, topOfCar - carBodyHeight);
}

void draw() {
  background(255);
  _______________________
}
```

What goes in the blank so this draws a car that is 100 pixels long and follows the mouse?
- A. `drawCar(100, mouseX, mouseY);`
- B. `drawCar(mouseX, mouseY);`
- C. `drawCar(mouseX, mouseY, 100);`

---

**Defining Simple Methods**

- `ReturnType Identifier (ParameterList) { Body }`
  - `ReturnType` is the type of value returned from the method/function.
  - `Identifier` is the name of the method/function.
  - `ParameterList` is a list of variables that will be used to pass information into the method. These are called the formal parameters.
  - `Body` is a list of statements and declarations describing the action performed by this method.

---

**Functions In Processing**

- Form of function definition in Processing
  ```java
  <return type> <name> (<param list>) {
  <body>
  }

  as in
  void draw_a_box(int x_pos, int y_pos) {
    rect(x_pos, y_pos, 20, 20);
    color pink() {
      return color(255, 200, 200);
    }
  }
  ```
  or
  ```java
  return color(255, 200, 200);
  ```
**Functions In Processing: Result**

- Functions that do something, but do not return a value, have `void` as their return type.
- Functions that return a value must say its type.

```java
void draw_a_box (int x_pos, int y_pos) {
  rect(x_pos, y_pos, 20, 20);
}
color pink () {
  return color(255, 200, 200);
}
```

**Parameters**

- Parameters are automatically declared (and initialized) on a call, and remain in existence as long as the function remains unfinished.
- When the function ends, the parameters vanish, only to be recreated on the next call.
- It is wise to choose parameter names that help you remember exactly what they mean.
  - `colorFlag`,
  - `dir` (for direction).

**Functions In Processing: Params**

- Parameters are the values used as input to the function; parameters are not required, but the parentheses are.
  - The type of each parameter must be given.

```java
void draw_a_box (int x_pos, int y_pos) {
  rect(x_pos, y_pos, 20, 20);
}
color pink () {
  return color(255, 200, 200);
}
```

**Functions In Processing: Return**

- A function returns its value with the `return` statement ... the stuff following return is the result.
- The function is done when it reaches return.

```java
void draw_a_box (int x_pos, int y_pos) {
  rect(x_pos, y_pos, 20, 20);
  color pink () {
    return color(255, 200, 200);
  }
}
```