Review Functions
Announcements

- Today: Guest Lecture on Human Computer Interaction and Design for Special Populations by Prof. Sri Kurniawan
- Co-Director of the Assistive Technologies Lab
- Functions review first
- Functions SUDOKU homework due Thursday
- Binary Arithmetic and Webpage HW assigned Thursday, due one week later
Sudoku GUI

The Sudoku board is a 9×9 array of cells. Alternating 3×3 cells are colored gray and strong lines separate all of the 3×3 cells, as follows. (More information at http://www.websudoku.com/.)
For loops (Repetition)

- Repeating commands is powerful:
  - Lightbot 2.0 used recursion, a function calling itself
  - Symbolic Lightbot prefixed a number, 2:Step
- Processing uses a `for` loop:

```java
void setup() {
  size(500,200);
  background(0);
  noStroke();
  smooth();
  fill(255);
  for (int i=0; i < 16; i++) {
    ellipse(100+25*i, 100, 15, 15);
  }
}
```
Using Functions

- Functions package up computation ... when do we use them? All the time.
- Write some simple code to achieve a goal ...

```java
void setup() {
    size(300, 300);
    background(102);
    noStroke();
    fill(255, 255, 0);
}

void draw() {
    rect(100, 100, 50, 50);
}
```
Package It … Make Position Vary

- To put the rectangle in different places, we “parameterize” the position, that is, use input to the function to place the rectangle

```java
void setup( ) {
    size(300, 300);
    background(102);
    noStroke( );
    fill(255,255,0);
}

void draw( ) {
    rec(100, 100);
}

void rec(float x, float y) {
    rect(x, y, 50, 50);
}
```
Now we quit thinking of drawing a rectangle, but now think of placing a 50x50 rectangle.
Example: Clock Timer. In sample codes.

- Draw digital timer elements
- Assemble elements into digits
- Light digit segments to create numbers
- Select number based on a digit
- Clocktimer.pde
Arguments Become Parameters

- Notice that if the DEFINITION has \( n \) parameters, the CALL needs \( n \) arguments
- The parameters and arguments correspond

Inside of the function, the parameter, e.g. \( xbase \), is declared and initialized to the corresponding argument, e.g. 80. Then, the definition uses it, e.g.

\[
\text{rect}(80, 40+10, 20, 40)
\]
More On Parameters …

**Parameters**: Customize each function call to a specific situation – they are the input to the function

- *Parameters* are the names of the input values used inside of the procedure body
- *Arguments* are the values from outside to be used for each of the parameters
Homework 10: Functions & For Loops

- void cell(int x, int y, int s, color tinto)
- void triple(int x, int y, int s, color tinto)
- Also functions for
  - block()
  - row()
  - cellarray()
Writing Functions

- Processing function definitions are typically listed after the standard blocks: setup(), draw(), mousePressed(), etc.

```java
void setup() {
    size(100, 100);
    background(0);
    noStroke();
}

void draw() {
    fill(255);
    hexa(20, 20);
}

void hexa(float xbase, float ybase) {
    rect(xbase, ybase+10, 20, 40);
    triangle(xbase, ybase+10, xbase+20, ybase+10, xbase+10, ybase);
    triangle(xbase, ybase+50, xbase+20, ybase+50, xbase+10, ybase+60);
}
```
Using Functions

- Once defined, functions can be called repeatedly ... it’s the point of writing them!

```c
void setup() {
  size(110, 100);
  background(0);
  noStroke();
}

void draw() {
  fill(255);
  hexa(20, 20);
  hexa(50, 20);
  hexa(80, 20);
}

void hexa(float xbase, float ybase) {
  rect(xbase, ybase+10, 20, 40);
  triangle(xbase, ybase+10, xbase+20, ybase+10, xbase+10, ybase);
  triangle(xbase, ybase+50, xbase+20, ybase+50, xbase+10, ybase+60);
}
```
Arguments Become Parameters

- Notice that if the DEFINITION has $n$ parameters, the CALL needs $n$ arguments
- The parameters and arguments correspond

```c
void draw( ) {
    fill(255);
    hexa(20, 40);
    hexa(50, 40);
    hexa(80, 40);
}

void hexa(float xbase, float ybase) {
    rect(xbase, ybase+10, 20, 40);
    triangle(xbase, ybase+10, xbase+20, ybase+10, xbase+10, ybase);
    triangle(xbase, ybase+50, xbase+20, ybase+50, xbase+10, ybase+60);
}
```

Inside of the function, the parameter, e.g. xbase, is declared and initialized to the corresponding argument, e.g. 80. Then, the definition uses it, e.g.

```c
rect(80, 40+10, 20, 40)
```
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 (Chao’s code),
  - dir (for direction).
- I chose xbase as the orientation point of the figure in the x direction, I use that name a lot and I know what it means.
Define hexa() and rexa()

- just_hexa_rexa.pde
- Parameterize the functions by a consistent position – upper left corner is good

```c
void draw() {
    fill(255);
    hexa(20, 40);
    rexa(30, 20);
}

void hexa(float xbase, float ybase) {
    rect(xbase, ybase+10, 20, 40);
    triangle(xbase, ybase+10, xbase+20, ybase+10, xbase+10, ybase);
    triangle(xbase, ybase+50, xbase+20, ybase+50, xbase+10, ybase+60);
}

void rexa(float xbase, float ybase) {
    triangle(xbase, ybase+10, xbase+10, ybase, xbase+10, ybase+20);
    rect(xbase+10, ybase, 40, 20);
    triangle(xbase+50, ybase, xbase+50, ybase+20, xbase+60, ybase+10);
}
```
Use hexa, rexa to make a Digit

- hexa_rexa_makeadigit.pde

```plaintext
void draw() {
    fill(255);
    digit(50, 20);
    digit(140, 20);
}

void hexa(float xbase, float ybase) {
    rect(xbase, ybase+10, 20, 40);
    triangle(xbase, ybase+10, xbase+20, ybase+10, xbase+10, ybase);
    triangle(xbase, ybase+50, xbase+20, ybase+50, xbase+10, ybase+60);
}

void rexa(float xbase, float ybase) {
    triangle(xbase, ybase+10, xbase+10, ybase, xbase+10, ybase+20);
    rect(xbase+10, ybase, 40, 20);
    triangle(xbase+50, ybase, xbase+50, ybase+20, xbase+60, ybase+10);
}

void digit(float xbase, float ybase) {
    hexa(xbase, ybase+10);    //left upper
    hexa(xbase, ybase+70);    //left lower
    rexa(xbase+10, ybase);    //mid horizontal
    rexa(xbase+10, ybase+60); //top horizontal
    rexa(xbase+10, ybase+120); //bot horizontal
    hexa(xbase+60, ybase+10); //right upper
    hexa(xbase+60, ybase+70); //right lower
}
```
Let There Be Light (and Dark)

- Define the illumination of the digit

hexa_rexa_makeadigit_addlight.pde

- Must declare two color variables, initialize to proper colors, use them in fill, and then check them

```pde
color dark, lite;

void setup( ) { 
  size(250, 180);
  background(0);
  stroke(0);
}

void draw( ) { 
  lite = color(255,185,0);
  dark = color(64, 48, 0);

  fill(dark);
  digit(50, 20);
  fill(lite);
  digit(140, 20);
}
```
Count In Lights: a function for each number

- Light up the digit for each number
- Count_in_lights.pde

```pde
void draw() {
  lite = color(255,185,0);
  dark = color(64, 48, 0);
  // one
  void one (float xbase, float ybase) {
    hexa(xbase+60, ybase);
    hexa(xbase+60, ybase+60);
  }
  // two
  void two (float xbase, float ybase) {
    rexa(xbase+10, ybase); // top horizontal
    hexa(xbase+60, ybase); // right upper
    rexa(xbase+10, ybase+60); // mid horizontal
    rexa(xbase+10, ybase+120); // bot horizontal
    hexa(xbase, ybase+60); // left lower
  }
}
```
Select A Number To Display

- Given an integer, display it in lights

```c
void sel(int n, float xbase, float ybase) {
    fill(lite);
    if (n == 0) {
        zero(xbase, ybase);
    }
    if (n==1) {
        one(xbase, ybase);
    }
    if (n==2) {
        two(xbase, ybase);
    }
    if (n==3) {
        three(xbase, ybase);
    }
    if (n==4) {
        four(xbase, ybase);
    }
    if (n==5) {
        five(xbase, ybase);
    }
    if (n==6) {
        six(xbase, ybase);
    }
}
```
Create a 3 Digit Display

```c
void three_digit(int n, float xbase, float ybase) {
    fill(dark);
    digit(50, 90);
    digit(140, 90);
    digit(260, 90);
    fill(lite);
    rect(xbase+185, ybase+125, 15, 15);
    sel(((n/100)%10, xbase, ybase);
    sel(((n/10)%10, xbase+90, ybase);
    sel(n%10, xbase+210, ybase);
}
```

Here’s The Action
Count up At The Frame Rate

```c
color dark, lite;
int i;

void setup( ) {
    size(400, 300);
    background(0);
    noStroke();
    frameRate(10);
}

void draw( ) {
    lite = color(255,185,0);
    dark = color(64, 48, 0);
    i = i + 1;
    three_digit(i, 50, 90);
}
```
Functional Abstraction: Layers of Functions

- Review What We Did

- The computation is ONLY drawing triangles and rectangles, but we don’t think of it that way ... to us, it’s a timer
Homework 10: Functions

- void cell(int x, int y, int s, color tinto)
- void triple(int x, int y, int s, color tinto)
- Also functions for
  - block()
  - row()
  - cellarray()