Topics for section today

Homework 10
functions
for loops
and
loading fonts
Draw the Sudoku board in Processing using for-loops and functions
Assignment 10 – Sudoku Board

Draw the Sudoku board in Processing using for-loops and functions
Assignment 10 – Sudoku Board

- 9 x 9 array of small cells
- 3 x 3 array of large cells
- The large cells are alternately colored gray or white
- Strong lines separate the large cells
- “Sudoku” title at the type
Put your name and assignment title in a comment at the top of your program!!!
Assignment Part 1. *Cell Array.* Begin with defining a function

```c
void cell(int x, int y, int s, color tinto)
```

where \(x\) and \(y\) are the positions of the upper left corner of a box, \(s\) is its size and \(tinto\) gives the color of the cell. The function simply draws a box filled with the proper color.

Using `cell( )`, you can build a function

```c
void triple(int x, int y, int s, color tinto)
```

where the parameters have the same meaning as before, though now \(x,y\) is the position of the upper left corner of three boxes in a row. Use a for loop in this function.
Functions

- You've been using functions since the Lightbot examples!

- You've been using functions in Processing since the very beginning!
Function calls

- A function call consists of the function name followed by a parenthesized list of parameter values, and completed with a semi-colon.

```
rect(x, y, width, height);
```

You should be familiar with this example of a function call to rect() from the Processing homework.
You should be familiar with these examples of function definitions from the Processing homework.
Parameter Values

In a function definition, the variables in the parameter list are defined, as in “int x”

```c
void draw_rectangle(int x, int y, int width, int height){
    fill(0,255,0);
    rect(x,y,width,height);
}
```

But in the function call, the variables represent distinct values or previously defined variables

```c
void draw(){
    background(255);
    draw_rectangle(50,50,100,100);
}
```
Writing your own functions

Say we wanted to write the function body for rect() from Processing.

How could you write it using the line() function, given that your function will have the same parameter values as rect(), which are int x, int y, int width and int height?

rect(x, y, width, height);
Step 1: Define the function

- Return type
- Function name
- Parenthesized list of defined parameters
- Opening and closing brackets

```c
void my_rectangle(int x, int y, int width, int height){
}
```

Notice that each variable is defined in the parameter list. Variables are defined, as in “int x”, in the parameter list of the function definition, but they are given distinct values in the function call.
Step 2: Fill in the function body

```c
void my_rectangle(int x, int y, int width, int height){
    stroke(0,0,255);
    strokeWeight(3);
    line(x,y,x+width,y);
    line(x+width,y,x+width,y+height);
    line(x+width,y+height,x,y+height);
    line(x,y+height,x,y);
}
```
Call the function in your program and you're done!

```cpp
void setup(){
  size(200,200);
}

void draw(){
  background(255);
  my_rectangle(50,50,100,100);
}

void my_rectangle(int x, int y, int width, int height){
  stroke(0,0,255);
  strokeWeight(3);
  line(x,y,x+width,y);
  line(x+width,y,x+width,y+height);
  line(x+width,y+height,x,y+height);
  line(x,y+height,x,y);
}
```
**for loops : anatomy**

```
for (int i = 0; i < 10; i++)
{
    //for-loop body
}
```

- **Keyword 'for'**
- **initialization**
- **test**
- **update**

Brackets to enclose the body of the for-loop, which will consist of whatever statements the programmer wishes to iterate over.
for loops : execution

How many iterations will this for-loop execute?

```cpp
for(int i = 0; i < 10; i++){
    //for loop body
}
```
for loops : execution

How many ellipses will be drawn after the loop has finished?

```java
for(int i = 0; i < 10; i++){
    ellipse(50+i*10,50,5,5);
}
```
Back to assignment 10 – Part 1
Create the set of functions that will help to draw the board.

Assignment Part 1. *Cell Array*. Begin with defining a function

```c
void cell(int x, int y, int s, color tinto)
```

where \(x\) and \(y\) are the positions of the upper left corner of a box, \(s\) is its size and \(tinto\) gives the color of the cell. The function simply draws a box filled with the proper color.

Using `cell()`, you can build a function

```c
void triple(int x, int y, int s, color tinto)
```

where the parameters have the same meaning as before, though now \(x, y\) is the position of the upper left corner of three boxes in a row. Use a `for` loop in this function.
Continue in this way, next building a $3 \times 3$ array of cells. The goal is the $9 \times 9$ array of cells created by the function `cellarray(x, y, s)`. [Hint: A good set of functions are: `cell()`, `triple()`, `block()`, `row()`, `cellarray()`.]
Fill in the functions cell() and triple()

//draw a single rectangle to represent a single cell of the sudoku board

void cell(int x, int y, int s, color tint) {
}

//draw three rectangles in a row to represent a row of three cells on the sudoku board

void triple(int x, int y, int s, color tint) {
Assignment 10 – Part 2

Draw the board.

Assignment Part 2. **Sudoku Board.** Write a function to draw the four thick black lines, given the position of the upper left corner of the cell array and the size of the cells as parameters. Package this function and the function from Part 1, into a function called “board” that has the position and size parameters.

```c
void board(int x, int y, int s){
    cellarray(x, y, s);
    fourlines(x,y,s,2);
}
```
**Drawing the dark lines**

Recall the required parameter values to draw a line in processing:

\[ \text{line}(x1,y1,x2,y2) \]

Calculate the \( x1, y1, x2, \) and \( y2 \) values for each of the four lines relative to the program variables \( x, y, \) and \( s. \)
Drawing the dark lines

(x+3*s, y)

(x+3*s, y+9*s)

Sudoku
Drawing the dark lines

Within the function you will use to draw the four lines, you will include the function call to line() with the appropriate parameter values. To make the lines thicker, you will use the function strokeWeight() with some value greater than 1 (the default).

```cpp
// draws the four thick lines on the sudoku board
void draw_four_lines(int x, int y, int s) {
    strokeWeight(3); // thicken the stroke weight for board lines
    line(x+3*s,y,x+3*s,y+9*s);
    line(...);
    line(...);
    line(...);
    strokeWeight(1); // reset the stroke weight for future drawings
}
```
Function: board()

Finally, you will put both the function call to cellArray() and draw_four_lines() within a single function titled board(), which you will invoke in the setup().

```cpp
void setup() {
  size(500, 500);
  background(255);
  s = (width-2*x)/9;
  board(x, y, s);  //draw the sudoku board
}

void board(int x, int y, int s) {
  cell_array(x, y, s, white_color);
  draw_four_lines(x, y, s);
}
Assignment 10 – Part 3
The board is now drawn. How do you interact with it?

Assignment Part 3. Identify Cell. We want to move the mouse pointer over the Sudoku board, and identify which cell it is over when the user clicks the mouse. When the click comes, we know the following information:

- `mouseX, mouseY` the position of the mouse (system variables)
- `x, y` upper left corner of the board, which is where we draw it
- `s` the size of the cells

and from these values you can figure out `cell_x, cell_y`, the position of the cell the user clicked on. For example, clicking on the center cell of the top row of squares should produce 4 for `cell_x` and 0 for `cell_y`. 
Assignment 10 – Part 3, continued

Write a `mousePressed()` function that figures out the values of `cell_x`, `cell_y`. These are the cell positions (0 through 8) that the mouse is over in the x- and y-directions. Someone else will need these later to write code to play the game of Sudoku. To prove that we’ve got this right, call your `cellarray()` function from `setup()` and then make your `draw()` function be:

```cpp
void draw() {
  if (mousePressed) {
    fill(255, 0, 0);
    ellipse(x + s * cell_x + s/2, y + s * cell_y + s/2, s/4, s/4);
  }
}
```

This draws a red dot in the cell the user clicked in.
Consider the content of `draw()`

```java
void draw() {
    if (mousePressed) {
        fill(255, 0, 0);
        ellipse(x+s*cell_x+s/2, y+s*cell_y+s/2, s/4, s/4);
    }
}
```

If `cell_x = 0` and `cell_y = 0`, where is the ellipse drawn?

If `cell_x = 4` and `cell_y = 3`, where is the ellipse drawn?
Consider the content of `draw()`

```java
void draw() {
    if (mousePressed) {
        fill(255, 0, 0);
        ellipse(x+s*cell_x+s/2, y+s*cell_y+s/2, s/4, s/4);
    }
}
```

If `cell_x = 0` and `cell_y = 0`, where is the ellipse drawn?

If `cell_x = 4` and `cell_y = 3`, where is the ellipse drawn?
Consider the content of draw()

```java
void draw() {
    if (mousePressed) {
        fill(255, 0, 0);
        ellipse(x+s*cell_x+s/2, y+s*cell_y+s/2, s/4, s/4);
    }
}
```

If cell_x = 0 and cell_y = 0, where is the ellipse drawn?

If cell_x = 4 and cell_y = 3, where is the ellipse drawn?
Think of the sudoku board as a coordinate system.

<table>
<thead>
<tr>
<th>cell_x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>cell_y</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Loading Fonts

Define a font variable at the top of your program.

```cpp
PFont my_font;
```

Then, inside your setup() function...

Use the function loadFont() to initialize the type of font.

```cpp
my_font = loadFont("Serif-48.vlw");
```

Use the function textFont() to initialize the size of the text. Leave the numerical parameter empty to use the default size.

```cpp
textFont(my_font, 32); or textFont(my_font);
```

Use the function text() to output words to the screen.

```cpp
text("I love CMPS10", x, y);
```
Did you get this error?

```cpp
PFont my_font;

void setup() {
    size(370, 340);
    background(255);

    my_font = loadFont("Serif-48.vlw");
    textAlign Center;
    fill(0);
    text("Sudoku",147,40);
}
```

```
Could not load font Serif-48.vlw. Make sure that the font has been copied to the data
at processing.core.PApplet.handleDraw(PApplet.java:1583)
at processing.core.PApplet.run(PApplet.java:1503)
at java.lang.Thread.run(Thread.java:680)
```
Go to:

Tools → Create Font
Find your font, and select OK
That's all for today. Questions?