1 □ Processing: Functions
   CST112

2 □ Structuring Programs (Modularity)
   ▪ Programmers often divide large applications into several modules within the program
   ▪ This is necessary due to the size and complexity of professional programs
   ▪ Hierarchies of structure allow the design to begin at the most general level ...
   ▪ And work to the more specific

3 □ To Find an Office in a Building
   ▪ A city
      ● has many districts or suburbs
         • each of which has many streets
            – each of which has many buildings
               » each of which has many offices

4 □ Top-Down Program Structure
   ▪ The problem
      ● has some major tasks
         • each of which has many subtasks
            – each of which has many subtasks
               ... and so on down as needed

5 □ Structure Using Functions  (Page 1)
   ▪ Complete applications in programming consist of a hierarchical collection of functions
   ▪ Working together they perform the entire programming task
   ▪ Most complex tasks can be subdivided into procedures in this way ("divide and conquer")

6 □ Structure Using Functions  (Page 2)
   ▪ Functions are written by a programmer to define specific tasks within the larger task...
      ● Each task should be clearly identifiable
   ▪ Statements defining the functions are written once in the application and may be called from more than one location in the program (reusability)
   ▪ Statements and variables are hidden from other functions in the program (information hiding)

7 □ Advantages of Using Functions
   ▪ Pre-written built-in functions (part of the Processing language/library) simplify program development
      ● E.g. line(), rect(), ellipse(), etc
   ▪ Each small piece can be written and tested separately
   ▪ Smaller functions usually are easier to understand
   ▪ In large professional systems individual programmers and programmer teams can build and share functions

8 □ The Function Call  (Page 1)
   ▪ The function is invoked by a function call
      ● The syntax specifies the function’s name
Programmers often divide large applications into several modules within the program.

Some possible arguments for the circle() and square() function calls are variables that are set speed by finding the absolute difference (over the assignment operator).

A program function is defined by a header (which names the function) and the function call.

The function call should use the returned value as it (Page 1)

Each small piece can be now written and the circle() and square() functions which use below(), right(), and Place a comment before each function header. Also if the radius is half the width of the output window, reset the fill color to random rgbRandom.

This is necessary due to the size and complexity of Complete applications in programming consist of a hierarchy of structure allow the design to begin at the most professional level ...

Example: Draw ―Stick Man‖

Statements and variables are data values in functions. Using the return value the key system variable.

The function definition is a means for communicating information. Each function has some major tasks which are communicated to other parts of the program.

Functions of types: Draw a ball inside an output window of 400 x 400

In function calls, there may be one or more values in parentheses (called arguments) which are passed to the called function (data “sharing”).

This is true for built-in Processing function calls as well as those defined by programmers.

<table>
<thead>
<tr>
<th>Function Call</th>
<th>(Page 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>functionName([argument1, argument2, ...]);</td>
<td></td>
</tr>
<tr>
<td>The function call will transfer control of the execution of the program to the location of the functionName.</td>
<td></td>
</tr>
<tr>
<td>Examples:</td>
<td></td>
</tr>
<tr>
<td>stickMan();</td>
<td></td>
</tr>
<tr>
<td>circle(x, y, radius);</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function Definition</th>
<th>(Page 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A program function is defined by a header (which names the function—also is called its signature) and the body enclosed in {braces}.</td>
<td></td>
</tr>
<tr>
<td>The format is similar to setup() or draw()</td>
<td></td>
</tr>
<tr>
<td>Format: type/void functionName([type parameter1, type parameter2, ...])</td>
<td></td>
</tr>
<tr>
<td>{ variableDeclarations } statements ...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function Definition</th>
<th>(Page 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples: void setup()</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>} void stickMan()</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

| Function Control |          |
| 1. The function call transfers control to the named sub function |          |
| 2. The body of that function is executed entirely |          |
| 3. At the conclusion of execution of the sub function, control returns to point in the initial function at which the sub function was called |          |

| Passing Arguments |          |
| In function calls, there may be one or more values in parentheses (called arguments) which are passed to the called function (data “sharing”). |          |
| This is true for built-in Processing function calls as well as those defined by programmers |          |
| Format: functionName(parameter1[, parameter2, ...]); |          |
| Examples: rect(x, y, w, l); |          |
circle(x, y, radius);

30 The Parameter List (Page 1)
- The parameter list is a comma-separated list of variable declarations in the function header
  ● The textbook calls this the “argument list”
  ● However for clarity and consistency with other languages, we will use the term parameter list
- Variables receive the values passed from the parameters in the function call

31 The Parameter List (Page 2)
- Format:
  type|void functionName( [type parameter1, type parameter2, ... ] )
  ● Each parameter in the list is a declared variable, declared separately with its own type (even if two or more parameters are the same type)
- Examples:
  void circle(int x, int y, int radius)
  void stickMan(int xCoord, int yCoord)

32 The Parameter List (Page 3)
- Parameters are local variables:
  ● A means for communicating information between the function call and the function itself
  ● When the function is called, the argument values are passed and assigned to the parameter variables in the function header

33 The Parameter List (Page 4)
- The number of arguments in the function call must be the same as the number of parameter variables in the called function header
- The variable type of each parameter in the function header must match that of the argument value passed to it (and in the same order)
- If the function is not passed any argument values, leave the parentheses empty:
  void setup()
  void resetRadius()

34 Function Call and Definition Examples
- Function calls:
  circle(75, 125, 50);
  circle(x, y, r);
  circle(mouseX, mouseY, radius);
- Function header (signature) and body
  void circle(int x, int y, int radius)
  {
   ...
  }

49 Passing Arguments “ByValue”
- Passing an argument by value means that a copy of the argument is stored as private in the called function at a separate RAM address ...
  ● Changing the value of the variable in the called function does not change the
original value

- The opposite of passing by value is passing by reference which means that the address of the values is passed to the called function ...
  - Changing the value of the variable in the called function will change the original value

**Return Values**

Programmer-defined functions:
1. May have arguments passed to them (the input)
2. Process that data (the code)
3. May have return values (the output from the function that is sent back to the location of the function call)

**Return Values**

Many functions “calculate” a return value (result) that is passed back to the calling function
- Statements using the keyword return are used to return a value from function to the call location
- Any number of return statements may be coded within the function, any of which:
  1. Terminate execution of function at that point
  2. Return control to the location at which it was called and passes the result back to the call

**Return Values**

Format for functions that return a result:
return expression;
- Returns expression value to calling function
- Examples:
  return 10;
  return x;
  return mouseX * mouseY;
  return random(0, 255);

**Return Value Example**

- Function call:
  int rgb = rgbRandom();
- Execution of all functions take precedence over the assignment operator
- Function definition:
  int rgbRandom()
  {
    return (int) random(0, 255);
  }

**Using the Return Value**

- The function call should use the returned value as part of another statement ...
  - Assignment statement, output statement, or some other data related operation
- Examples:
  r = rgbRandom();
  println( rgbRandom() );
if (rgbRandom() < 127) ...
fill( rgbRandom(), rgbRandom(), rgbRandom() );

55 Invalid Function Calls
- Therefore functions that return a value should not stand alone
- The circle(int, int, int) function does not return a value so the following is valid:
circle(mouseX, mouseY, rad);
- The rgbRandom() function does return a value so the following is invalid:
rgbRandom();

58 Function Types
- Functions that return a value have a type
  ● A function’s type must be the same as the type of the value that is returned
- Format:
  type functionName( [parameterList] )
- Example:
  int rgbRandom()
- If a value is not returned, the return type is void
  void circle(int xCoord, int yCoord, int radius)

59 The key System Variable
- A Processing system variable that stores the last character that was used on the keyboard (either keyed or released)
- Special system constants that can be tested for are BACKSPACE, TAB, ENTER, ESC, etc.
- Examples:
  if (key == 'A')
    ● Differentiates between upper and lower cases
  if (key == ENTER) // The Enter key

64 The sq and sqrt Functions
- These are two Math functions:
  ● The sq() function returns the square of a value, (the product of multiplying it by itself), i.e.
    square = sq(9); // square will be 81
  ● The sqrt() function returns the square root of a value, i.e.
    root = sqrt(9); // root will be 3

65 Using the Return Value
- A function call that returns a value can be used to represent values in a larger expression (not just in assignment statements)
- Examples:
  println( rgbRandom() );
circle(x, y, radius(x, y ));
fill( rgbRandom(), rgbRandom(), rgbRandom() );

73 Final Review of Function Calls
- So remember that there are three distinct elements involved in calling functions:
  1) The function call transfers control to the called function (and automatically
transfers back to the call location when done executing)
2) One or more argument values may be passed to parameter variables that are defined in the called function’s header
3) A value (the result of function’s processing) may be returned after its execution