CS 101
Functions

Lecture 15
Key Processing language features so-far

- Basic color/shapes drawing
- Variables
- For-loops
- If-statements
Functions

- In the next few days, we’ll be talking about **Functions**
- A **Function** is a way to “save” a series of instructions for (re-)use
- Let’s start with a motivating example
What does it do?

```java
void setup() {
  size(500, 500);
  background(100, 150, 100);
}

int personX = 50;
int personY = 50;

void draw() {
  noStroke();
  // head
  fill(130, 100, 80);
  rect(personX + 0, personY + 0, 200, 200);
  // eyes
  fill(100, 240, 100);
  ellipse(personX + 50, personY + 50, 40, 40);
  ellipse(personX + 150, personY + 50, 40, 40);
  // mouth
  fill(200, 100, 100);
  ellipse(personX + 100, personY + 150, 100, 30);
  // nose
  fill(80, 60, 40);
  triangle(personX + 100, personY + 120, personX + 70,
           personY + 120, personX + 100, personY + 50);
}
```
Questions

What would we do if we wanted to draw multiple faces on the screen?

How to do this? What code-constructs do we know about for repetition?
Question

How could the code be changed to produce this, for example?
What would we do if we wanted to draw multiple faces on the screen?

What code-constructs do we know about for repetition?

```cpp
for (int i = 0 ; i < 600 ; i += 220) {
    // head
    fill(130, 100, 80);
    rect(i + personX + 0, personY + 0, 200, 200);
    // eyes
    fill(100, 240, 100);
    ellipse(i + personX + 50, personY + 50, 40, 40);
    ellipse(i + personX + 150, personY + 50, 40, 40);
    // mouth
    fill(200, 100, 100);
    ellipse(i + personX + 100, personY + 150, 100, 30);
    // nose
    fill(80, 60, 40);
    triangle(i + personX + 100, personY + 120, i + personX + 70
             personY + 120, i + personX + 100, personY + 50);
}```
But what about this?

How would you implement this?
Functions

- The best way to do this is to use a function
- As stated before: A function is a code-construct in which we can “save” a sequence of program instructions and tell processing to execute them at any time we want
- We have seen built-in functions such as rect() and ellipse()
- You can also define your own functions
Defining a Function

- When we define a function, we are telling Processing the instructions that we want the function to execute when we call it.
- This looks like:

```java
void functionName() {
    // lines of code
    /// ...
}
```
void functionName() {
    // lines of code
    // ...
}

The function name (in camelCase too)

void functionName() {
    // lines of code
    ...
}

For now, just know that you should start all of your function definitions with “void”

For now, empty parentheses come right after name

The lines of code that will be run when called

Open and close curly-braces (just like for-loops, if-statements)
Calling a Function

- Once a function is **defined** it can be **called**
- Calling a function causes the lines of code within it to be executed
- Calling a function is simple:

```javascript
functionName();
```
Calling a Function

An example . . .

```java
void drawPatternA() {
    background(200, 100, 100);
    fill(0, 255, 100);
    for (int i = 10 ; i < 600; i += 100) {
        triangle(i, 10, i+90, 190, i, 190);
    }
}

void drawPatternB() {
    background(100, 100, 200);
    fill(0, 100, 155);
    for (int i = 50 ; i < 600; i += 100) {
        ellipse(i, 100, 80, 80);
    }
}

void drawPatternC() {
    background(100, 200, 100);
    fill(0, 100, 255);
    for (int i = 100 ; i < 600; i += 200) {
        ellipse(i, 100, 180, 180);
    }
}

void setup() {
    size(600, 200);
    background(100, 150, 100);
}

void draw() {
    noStroke();
    if (mouseX < 200) {
        drawPatternA();
    } else if (mouseX < 400) {
        drawPatternB();
    } else {
        drawPatternC();
    }
}
```
Drawing a house

Implement the drawHouse Function, so that the canvas appears like this, starting with the below code:

```java
void setup () {
    size(400, 400);
}

void draw() {
    drawHouse();
}
```
Drawing a house

```java
void setup () {
  size(400, 400);
}

void draw() {
  drawHouse();
}

void drawHouse() {
  strokeWeight(2);
  fill(50, 200, 255);
  triangle(200, 100, 100, 150, 300, 150);
  rect(100, 150, 200, 100);
  rect(150, 175, 50, 75);
}
```
Functions and Faces

- What if we want more than one house.
- How can we use the function to put a house “wherever we want”?
- How would we change the x- and y-coordinates for each call to drawHouse()
Functions and Faces

- One solution: change `drawHouse()` to use variables `xPos` and `yPos`
- Assign to `xPos` and `yPos` outside of the function, then call the function
  - assign to `xPos`
  - assign to `yPos`
  - call `drawHouse()`
  - change `xPos` and `yPos`
  - call `drawHouse()` again
Functions and Houses

- Now think back to the faces problem
- How can we use a function to put faces “wherever we want”? 
int faceX = -20;
int faceY = 100;

void drawFace() {
  // head
  fill(130, 100, 80);
  rect(faceX + 0, faceY + 0, 200, 200);
  // eyes
  fill(100, 240, 100);
  ellipse(faceX + 50, faceY + 50, 40, 40);
  ellipse(faceX + 150, faceY + 50, 40, 40);
  // mouth
  fill(200, 100, 100);
  ellipse(faceX + 100, faceY + 150, 100, 30);
  // nose
  fill(80, 60, 40);
  triangle(faceX + 100, faceY + 120,
           faceX + 70, faceY + 120,
           faceX + 100, faceY + 50);
}

void draw() {
  noStroke();
  faceX = -20;
  faceY = 100;
  drawFace();
  faceX = 100;
  faceY = 50;
  drawFace();
  faceX = 300;
  faceY = 360;
  drawFace();
  faceX = 620;
  faceY = 400;
  drawFace();
  faceX = 500;
  faceY = 100;
  drawFace();
  faceX = -30;
  faceY = 500;
  drawFace();
  faceX = 320;
  faceY = -120;
  drawFace();
}
Functions and Faces

- It works!
  - Annoying having to change the value of the variables each time the program runs
- There is a better way . . .
Parameters

- A **parameter** is a special type of variable that is sent to a function when it is called
  - Can be used only within the function it is associated with
- For example, consider calling the function `rect()`:
  - `rect(20, 40, 50, 100);`
  - The numbers 20, 40, 50 and 100 are parameters
- When we define a function, we define the types and names of the parameters
void functionName(int p1) {
    // lines of code
    ...
}

This is a function definition with one parameter

```c
void functionName(int p1) {
    // lines of code
    // ...
}
```

Parameters go in-between the parentheses:

- `void`: The return type of the function (void means the function does not return a value).
- `functionName`: The name of the function.
- `int`: The type of the parameter `p1`.
- `p1`: The parameter name.

The `p1` variable can only be used within the function body.

Give the type and the name, but don’t assign a value to the variable.
void drawCenterCircle(int p1) {
    fill(100, 200, 100);
    ellipse(width/2, height/2, p1, p1);
}

do not draw(...);
void drawCenterCircle(int p1) {
  fill(100, 200, 100);
  ellipse(width/2, height/2, p1, p1);
}

void draw() {
  drawCenterCircle(200);
}

The argument 200 is passed to the function parameter diameter when it is called.

Only usable within the function.
```java
void drawCenterCircle(int p1) {
  fill(100, 200, 100);
  ellipse(width/2, height/2, p1, p1);
}

void draw() {
  drawCenterCircle(200);
}
```

The argument 200 is passed to the function parameter diameter when it is called.

Only usable within the function.
void drawCenterCircle(int p1) {
  fill(100, 200, 100);
  ellipse(width/2, height/2, p1, p1);
}

void draw() {
  drawCenterCircle(200);
}

The argument 200 is passed to the function parameter diameter when it is called.

Only usable within the function.
void drawCenterCircle(int p1) {
  fill(100, 200, 100);
  strokeWeight(5);
  ellipse(width/2, height/2, p1, p1);
}

void draw() {
  drawCenterCircle(200);
  drawCenterCircle(100);
  drawCenterCircle(50);
}
void drawCenterCircle(int p1) {
  fill(100, 200, 100);
  strokeWeight(5);
  ellipse(width/2, height/2, p1, p1);
}

void draw() {
  drawCenterCircle(200);
  drawCenterCircle(100);
  drawCenterCircle(50);
}

When drawCenterCircle is called the first time, the diameter parameter is given the argument 200
void drawCenterCircle(int p1) {
    fill(100, 200, 100);
    strokeWeight(5);
    ellipse(width/2, height/2, p1, p1);
}

void draw() {
    drawCenterCircle(200);
    drawCenterCircle(100);
    drawCenterCircle(50);
}

When `drawCenterCircle` is called the second time, the diameter parameter is given the argument 100
```java
void drawCenterCircle(int p1) {
  fill(100, 200, 100);
  strokeWeight(5);
  ellipse(width/2, height/2, p1, p1);
}

void draw() {
  drawCenterCircle(200);
  drawCenterCircle(100);
  drawCenterCircle(50);
}
```

When `drawCenterCircle` is called the third time, the diameter parameter is given the argument 50.
Drawing a house

Modify the house-drawing code so that the y coordinate (vertical position) can be modified with the parameter. The below code should produce the pictured canvas.

```java
void setup () { 
  size(400, 400);
}

void draw() { 
  drawHouse(25);
  drawHouse(225);
}
```
void setup () {
  size(400, 400);
}

void draw() {
  drawHouse(25);
  drawHouse(225);
}

void drawHouse(int y) {
  strokeWeight(2);
  fill(50, 200, 255);
  triangle(200, y, 100, y+50, 300, y+50);
  rect(100, y+50, 200, 100);
  rect(150, y+75, 50, 75);
}
Back to the face program

- Can we make a program do this with just one parameter?
- Could control X, or Y, but not both
- We need two parameters
void functionName(int x, int y, int d) {
    // lines of code
    // ...
}

This is a function definition with multiple parameters

```
void functionName(int x, int y, int d) {
    // lines of code
    // ...
}
```

All parameters only usable within the function body

Again, parameters go in-between the parentheses

Separate each parameter with a comma
void drawCircle(int xPos, int yPos, int diameter) {
  fill(100, 200, 100);
  strokeWeight(5);
  ellipse(xPos, yPos, diameter, diameter);
}

void draw() {
  drawCircle(50, 100, 30);
  drawCircle(40, 30, 50);
  drawCircle(101, 110, 120);
}
void drawCircle(int xPos, int yPos, int diameter) {
    fill(100, 200, 100);
    strokeWeight(5);
    ellipse(xPos, yPos, diameter, diameter);
}

draw();

drawCircle(50, 100, 30);
drawCircle(40, 30, 50);
drawCircle(101, 110, 120);
void drawCircle(int xPos, int yPos, int diameter) {
  fill(100, 200, 100);
  strokeWeight(5);
  ellipse(xPos, yPos, diameter, diameter);
}

void draw() {
  drawCircle(50, 100, 30);
  drawCircle(40, 30, 50);
  drawCircle(101, 110, 120);
}
```java
void drawCircle(int xPos, int yPos, int diameter) {
  fill(100, 200, 100);
  strokeWeight(5);
  ellipse(xPos, yPos, diameter, diameter);
}

void draw() {
  drawCircle(50, 100, 30);
  drawCircle(40, 30, 50);
  drawCircle(101, 110, 120);
}
```
Modify the house-drawing code so that there is a parameter for x and y.

The below code should produce the pictured canvas

```java
void setup () {
  size(400, 400);
}

void draw() {
  drawHouse(120, 25);
  drawHouse(200, 225);
}
```
Drawing a house (again)

```cpp
void setup () {
  size(400, 400);
}

void draw() {
  drawHouse(120, 25);
  drawHouse(200, 225);
}

void drawHouse(int x, int y) {
  strokeWeight(2);
  fill(50, 200, 255);
  triangle(x, y, x-100, y+50, x+100, y+50);
  rect(x-100, y+50, 200, 100);
  rect(x-50, y+75, 50, 75);
}
```
Back to the face program

• What would the face function look like to control the x and y position of the face?
Back to the face program

- Knowing this, we can create a function to draw a face
- Then, call it a bunch of times with whatever X and Y coordinates we want

```cpp
/*
 * Function to draw a 200x200 square face
 * The first parameter is the X position
 * The second parameter is the Y position
 */
void drawFace(int faceX, int faceY) {
  // head
  fill(130, 100, 80);
  rect(faceX + 0, faceY + 0, 200, 200);
  // eyes
  fill(100, 240, 100);
  ellipse(faceX + 50, faceY + 50, 40, 40);
  ellipse(faceX + 150, faceY + 50, 40, 40);
  // mouth
  fill(200, 100, 100);
  ellipse(faceX + 100, faceY + 150, 100, 30);
  // nose
  fill(80, 60, 40);
  triangle(faceX + 100, faceY + 120,
           faceX + 70, faceY + 120,
           faceX + 100, faceY + 50);
}
```
Global vs Local variables

- A **global variable** is a variable that is declared outside of a function.
- Global variables can be accessed and used by all functions in your program.
- When you modify a global variable, the effect is “global”.

```cpp
// variables to determine the X and Y positions of player one of the game
int playerOneX = 100;
int playerOneY = 200;

void setup() {
  size(300, 300);
  frameRate(30);
  playerOneX = 200;
}

void draw() {
  rect(playerOneX, playerOneY, 30, 30);
  playerOneX = playerOneX + 5;
}
```
Global vs Local variables

- A **global variable** is a variable that is declared outside of a function.
- Global variables can be accessed and used by all functions in your program.
- When you modify a global variable, the effect is "global".

```java
void setup() {
  size(300, 300);
  frameRate(30);
  playerOneX = 200;
}

// variables to determine the X and Y positions of player one of the game
Int playerOneX = 100;
Int playerOneY = 200;

void draw() {
  rect(playerOneX, playerOneY, 30, 30);
  playerOneX = playerOneX + 5;
}
```
Global vs Local variables

- **A global variable** is a variable that is declared outside of a function.
- Global variables can be accessed and used by all functions in your program.
- When you modify a global variable, the effect is “global”.

```java
void setup() {
  size(300, 300);
  frameRate(30);
  playerOneX = 200;
}

void draw() {
  rect(playerOneX, playerOneY, 30, 30);
  playerOneX = playerOneX + 5;
}

// variables to determine the X and Y positions of player one of the game
Int playerOneX = 100;
Int playerOneY = 200;
```
Global vs Local variables

- A **local variable** is a variable that is declared within a function
- Local variable only accessible in the function
- Parameters count too

```java
void setup() {
  // cannot use width here
  size(300, 300);
  frameRate(30);
  playerOneX = 200;
}

void draw() {
  checkMousePosition();
  int width = mouseX / mouseY;
  rect(playerOneX, playerOneY, width, width);
  playerOneX = playerOneX + 5;
}

void checkMousePosition() {
  // cannot use width here
  if (mouseX > mouseY) {
    background(255, 0, 0);
  } else {
    background(0, 0, 255);
  }
}
```
What do these programs do?

```java
void setup () {
    size(500, 200);
}

void draw() {
    int offset = 0;
    background(100, 100, 100);
    rect(offset, 25, 25, 150);
    offset += 2;
}
```

```java
void setup () {
    size(500, 200);
}

int offset = 0;

void draw() {
    background(100, 100, 100);
    rect(offset, 25, 25, 150);
    offset += 2;
}
```
What do these programs do?
void setup () {
  size(500, 300);
}

int position = 0;

void draw() {
  background(100, 100, 100);
  for (int i = 5; i < width; i += 50) {
    rect(i, position, 40, 40);
  }
  position += 2;
}
What does this program do?
How about these two modified versions?

```cpp
void setup () {
    size(500, 300);
}

void draw() {
    int position = 0;
    background(100, 100, 100);
    for (int i = 5; i < width; i += 50) {
        rect(i, position, 40, 40);
    }
    position += 2;
}
```

```cpp
void setup () {
    size(500, 300);
}

void draw() {
    int position = 0;
    background(100, 100, 100);
    for (int i = 5; i < width; i += 50) {
        rect(i, position, 40, 40);
        position += 25;
    }
}
```
How about these two modified versions?
What do these two programs do?

```cpp
void setup () {
  size(500, 500);
  stroke(0, 200, 255);
  strokeWeight(5);
}

int position = 0;

void draw() {
  background(100, 100, 100);
  drawStuff(position);
  position += 2;
}

void drawStuff(int size) {
  rect(20, 20, size, size);
  rect(30, 30, size-20, size-20);
}
```

```cpp
void setup () {
  size(500, 500);
  stroke(0, 200, 255);
  strokeWeight(5);
}

int position = 0;

void draw() {
  background(100, 100, 100);
  drawStuff(position);
  position += 2;
}

void drawStuff(int size) {
  rect(20, 20, size, size);
  size -= 20;
  rect(30, 30, size, size);
}
```
What do these two programs do?
Materials

- Required Materials
  - GSWP Chapter 9