Lecture F1
While Loop Tutorial

Computing and Art : Nature, Power, and Limits
CC 3.12: Fall 2007

Functionalia

Instructor

Chipp Jansen, ***chipp@sci.brooklyn.cuny.edu***

Course Web Page

`http://www.sci.brooklyn.cuny.edu/~chipp/cc3.12/`

- **HW E**: **DUE** Wednesday, Nov 14th, 11:59 pm
- Wednesday Open Lab 9:30 - 12:00 (but NOT Office Hours)

Office Hours | Extra Help

My Office Hours :

Mondays 12:30 to 1:30 - basement Ingersoll 0317N

Virtual: Tuesdays 9 to 11pm - AIM: chippbot

Bridges Student Resource Center : 0317N

Monday	1:30 to 5pm
Tuesday	1:30 to 3:30
Wednesday	1:30 to 3:30
Thursday	12:45 - 1:45, 2:15 - 3:15

Functionalia

Today:

- Iteration and the While Loop.

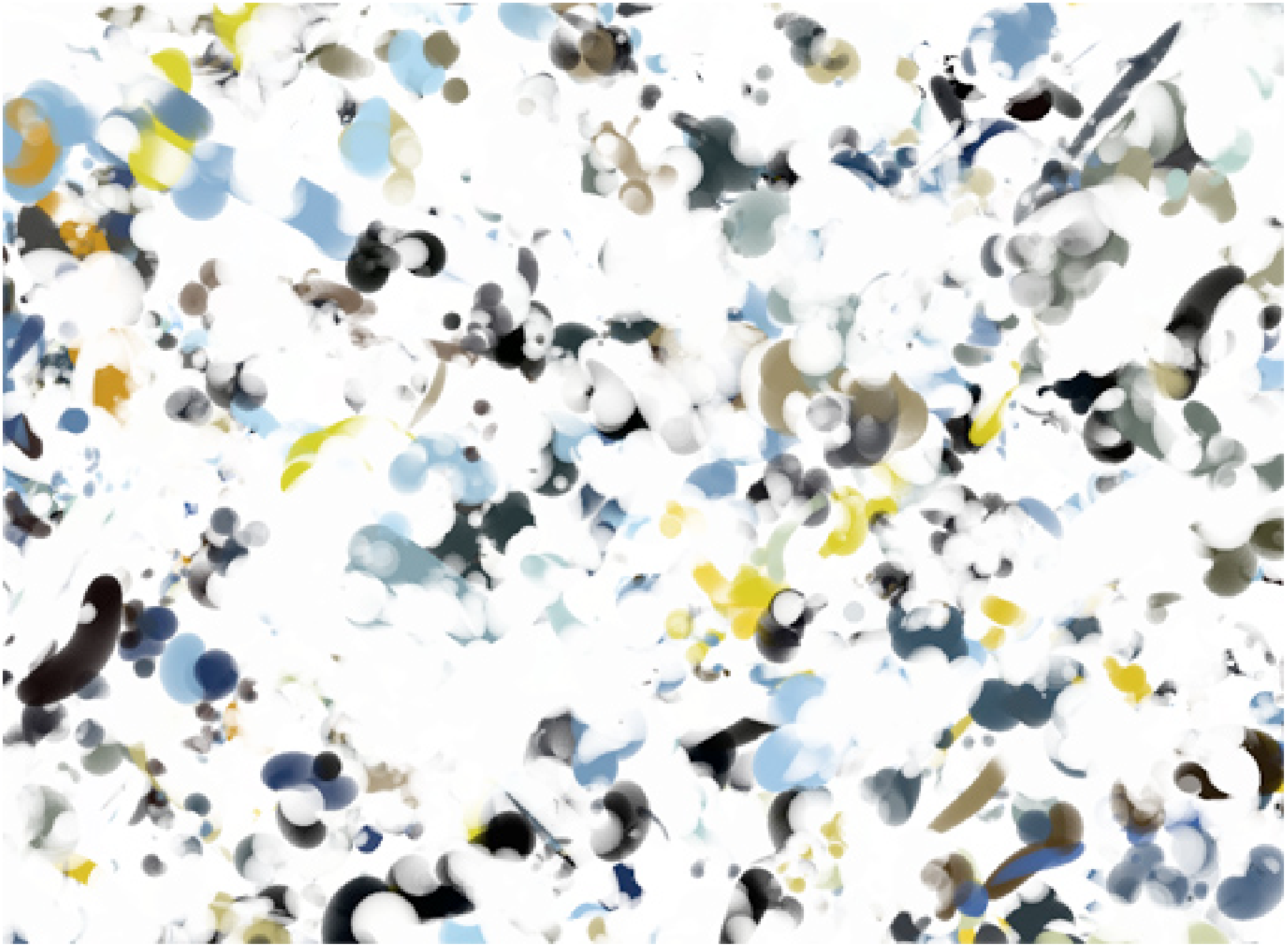
Iteration

Iteration means doing something more than once perhaps doing somethings over and over ... and over again.

Iteration allows us to tell the computer to do repetitive task with explicitly telling the computer what to do.

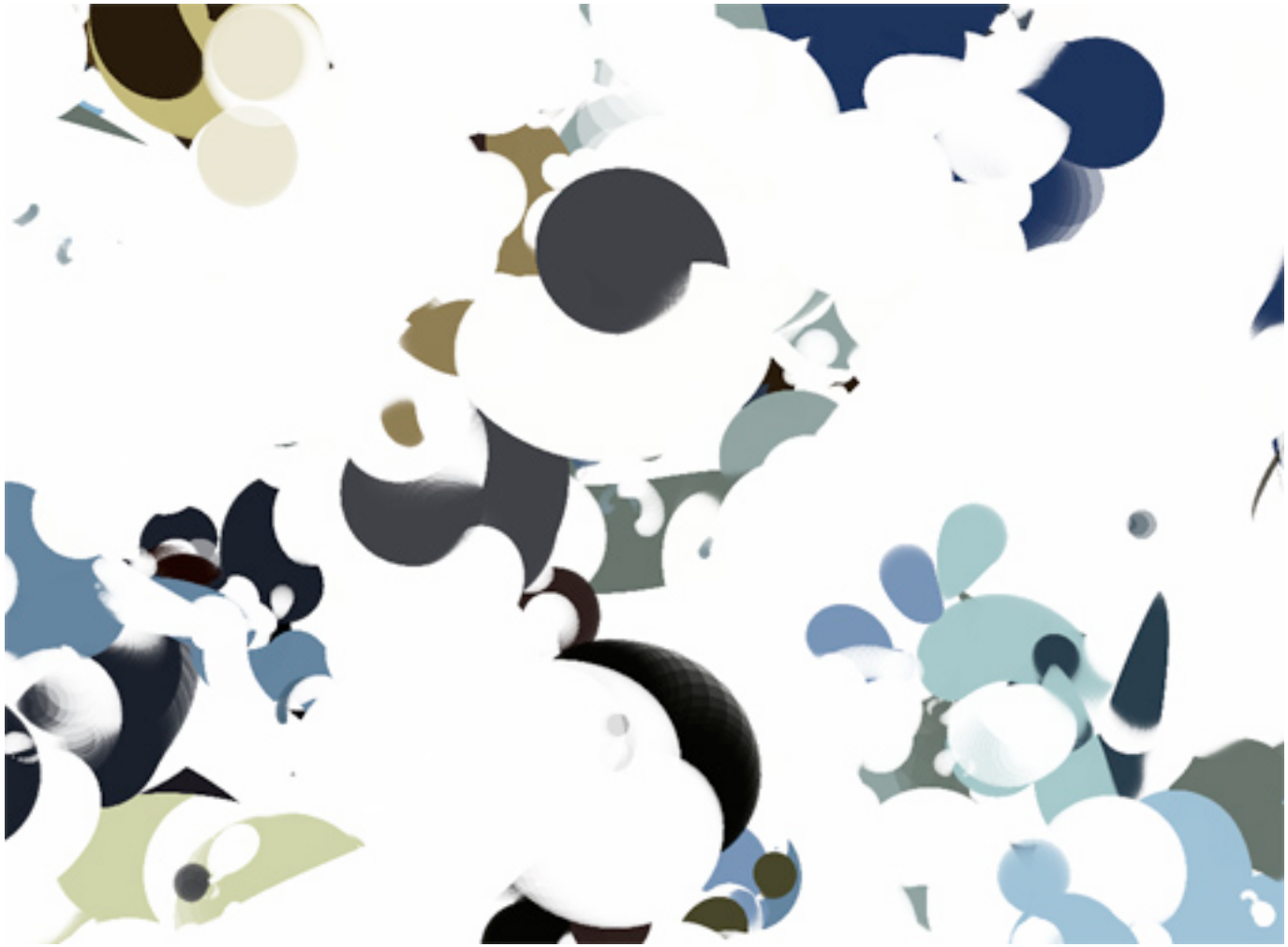
Example: Tell the computer to draw 100 ellipses in a row, which having to write the ellipse function 100 times.

Iteration allows for Complex Repetition



Variety is possible with Iteration

Shapes are basically circular, but vary in size, color, and palcement.



Loops

On a computer, we implement Iteration through Loops.

Two kinds of loops:

infinite loops: repeat indefinatly until you quit the program, or turn off the computer. The draw() function is example of something that loops infinitely.

conditional loops: repeats while a certain condition is true. These loops allow you to do something for a certain number of times.

In Processing, we will use a **`while`** loop to repetitively draw a shape.

There are other loops in programming languages:

- for loops
- do... while loops
- for each loops

```
// Add variables here

void setup() {

   size(600, 600);

}


void draw() {


}
```

1. Start with the Start template.

```
// Add variables here

int x;

void setup() {

  size(600, 600);

  x = 0;

  println("Value of x " + x);

}


void draw() {


}
```

2. We will start by adding a variable (**x**) to use in our loop.

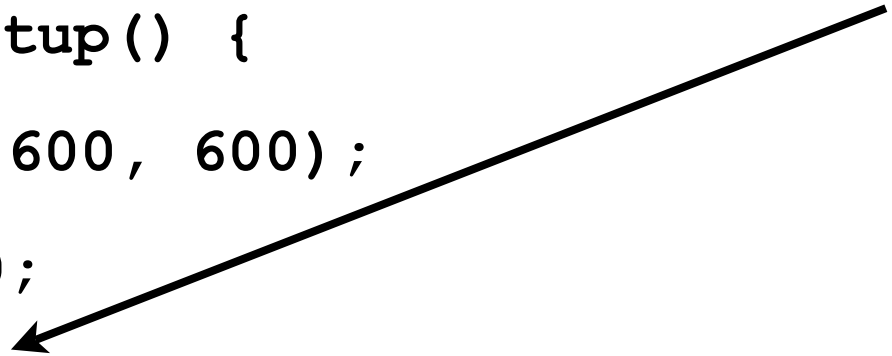
This variable is called the **loop variable**.

Loops use variables to keep track of progress, and to know when to stop.

```
// Add variables here
int x;
void setup() {
  size(600, 600);
  x = 0;
  while(x < 10) {
    println("Value of x " + x);
    x = x + 1;
  }
}

void draw() {
}
```

3. Let us put the println statement into a while loop to repetitively print the value of x out.

```
// Add variables here

int x;

void setup() {

  size(600, 600);

  x = 0;

  while(x < 10) {

    println("Value of x " + x);

    x = x + 1;

  }

}


void draw() {

}
```



4. The structure is the same as the if() conditional.

Starts with the word "`while`"

```
// Add variables here

int x;

void setup() {
  size(600, 600);
  x = 0;
  while(x < 10) {
    println("Value of x " + x);
    x = x + 1;
  }
}


void draw() {
}
```

5. Then there is a conditional. Just like the if() statement.

```
// Add variables here

int x;

void setup() {
  size(600, 600);
  x = 0;
  while(x < 10) {
    println("Value of x " + x);
    x = x + 1;
  }
}


void draw() {
}
```

6. While this conditional is **`true`**, what follows in the curly braces runs over and over again. Until the conditional becomes **`false`**.

```
// Add variables here

int x;

void setup() {
  size(600, 600);
  x = 0;
  while(x < 10) {
    println("Value of x " + x);
    x = x + 1;
  }
}


void draw() {
}
```

7. What is printed out? How do the values of **x** change through the loop?

```
// Add variables here

int x;

void setup() {
  size(600, 600);
  x = 0;
  while(x < 10) {
    println("Value of x " + x);
    x = x + 1;
  }
}


void draw() {
}
```

8. Notice every time through the loop the value of x is increased by one. At what value of x does the loop stop?

```
// Add variables here

int x;

void setup() {

  size(600, 600);

  x = 0;

  while(x < 10) {

    println("Value of x " + x);

    x = x + 2;

  }

}


void draw() {

}
```

9. What happens when the value of x is increased is changed to 2?

How many times does the loop run?

```
// Add variables here

int x;

void setup() {
  size(600, 600);
  x = 0;
  while(x < 10) {
    println("Value of x " + x);
    ellipse(x, 100, 20, 20);
    x = x + 2;
  }
}

void draw() {
}
```

10. Let's add an ellipse to the while loop.

We will use the loop variable **x** in place of the x parameter of the ellipse.

What happens? How many ellipses are drawn on the screen?

```
// Add variables here

int x;

void setup() {
  size(600, 600);
  x = 0;
  while(x < 600) {
    println("Value of x " + x);
    ellipse(x, 100, 20, 20);
    x = x + 2;
  }
}

void draw() {
}
```

11. In the previous step: The ellipses were drawn from the x value range of 0 to 10. The range of values that the variable x contained.

Let us increase the range by increasing the limit in the conditional to 600 (the width of the screen).

How many ellipses will be drawn?

```
// Add variables here
int x;
void setup() {
  size(600, 600);
  x = 0;
  while(x < 600) {
    println("Value of x " + x);
    ellipse(x, 100, 20, 20);
    x = x + 100;
  }
}
void draw() {
}
```

12. The ellipses are drawn on top of each other. We can increase the spacing between the ellipses by increasing the value that the x variable is increased by every time through the while loop.

Let us increase the value to 100, how many ellipses will be drawn?

```
// Add variables here
int x;
void setup() {
  size(600, 600);
  x = 0;
  while(x < 600) {
    println("Value of x " + x);
    fill(0, x, 0);
    ellipse(x, 100, 20, 20);
    x = x + 100;
  }
}
void draw() {
}
```

13. We can use the loop variable in other functions as well in the loop.
Changing the fill color of the ellipse for example.

What is the color range for these ellipses?

```
// Add variables here
int x;
void setup() {
  size(600, 600);
  x = 0;
  while(x < 600) {
    println("Value of x " + x);
    fill(0, x/3, 0);
    ellipse(x, 100, 20, 20);
    x = x + 100;
  }
}
void draw() {
}
```

14. Since the range of color values are 0 to 255. And our variable varies from 0 to 600, we can use division to narrow the fill value.

We can divide x by 3, and pass the resulting value to the fill() function.

Note: this does NOT change the value of x for the loop.

```
// Add variables here
int x;
void setup() {
  size(600, 600);
  x = 0;
  while(x < 600) {
    println("Value of x " + x);
    fill(0, x/3, 0);
    ellipse(x, 100, 20, 20);
    x = x + 100;
  }
}
void draw() {
}
```

15. Other mathematical operations that are possible.

Math	Code
add	+
subtract	-
divide	/
multiply	*

```
// Add variables here
int x;
void setup() {
  size(600, 600);
  x = 0;
  while(x < 600) {
    println("Value of x " + x);
    fill(0, x/3, 0);
    ellipse(x, 100, 20, 20);
    x = x + 100;
  }
}
void draw() {
}
```

16. Up until this point, the while loop has been in the setup() function. Remember, that setup() occurs only once, at the beginning of the program.

Let's see what happens when we move the while loop to the draw function...

```
// Add variables here
int x;
void setup() {
  size(600, 600);
}

void draw() {
  x = 0;
  while(x < 600) {
    println("Value of x " + x);
    fill(0, x/3, 0);
    ellipse(x, 100, 20, 20);
    x = x + 100;
  }
}
```

17. Notice that everything except for the size() function was moved to the draw function (even setting the variable x to 0: x = 0;)

What happens now when you run the program?

What is printed out?

```
// Add variables here
int x;
void setup() {
  size(600, 600);
}

void draw() {
  x = 0;
  while(x < 600) {
    println("Value of x " + x);
    fill(0, x/3, 0);
    ellipse(x, 100, 20, 20);
    x = x + 1 + mouseX;
  }
}
```

18. Now that the draw function runs repetitively, we can add the use of the mouse position variables: mouseX and mouseY to the while loop.

Change the value that x is incremented by to use the variable. (**Note**: you always want the value to increase by at least 1, why?)

```
// Add variables here
int x;
void setup() {
  size(600, 600);
}

void draw() {
  background(255);
  x = 0;
  while(x < 600) {
    println("Value of x " + x);
    fill(0, x/3, 0);
    ellipse(x, 100, 20, 20);
    x = x + 1 + mouseX;
  }
}
```

19. There will be trails of the ellipse as it moves about the screen. We can remove what has been drawn in every draw function by putting the background() function at the top of the draw function.

```
// Add variables here
int x;
void setup() {
  size(600, 600);
}

void draw() {
  background(255);
  x = 0;
  while(x < 600) {
    println("Value of x " + x);
    fill(0, x/3, 0);
    ellipse(x, 100, 20, 20);
    x = x + 1 + mouseX/10;
  }
}
```

20. Notice that the row of ellipses expands very quickly as you move the mouse from the left to the right-hand side of the screen.

We can limit the effect that the **`mouseX`** function has on the value of x by dividing it by a number.

Normally, mouseX has a range of 0 to the width of the sketch (600 in this case). Dividing it by 10, will give it a range of 0 to 60).

```
// Add variables here
int x;
void setup() {
  size(600, 600);
}

void draw() {
  background(255);
  x = 0;
  while(x < mouseX) {
    println("Value of x " + x);
    fill(0, x/3, 0);
    ellipse(x, 100, 20, 20);
    x = x + 20;
  }
}
```

21. Another variation is to have the variable x increment back at a steady rate (x = x + 20;)

Then replace the limit in the conditional to the variable mouseX.

What happens as you move the mouse along the screen.

How are the values of x changing? (Look in the console)

HW F

HW F Part 1 continues to work with the while loop.

- **HW F**: **DUE** Wednesday, Nov 28th, 11:59 pm