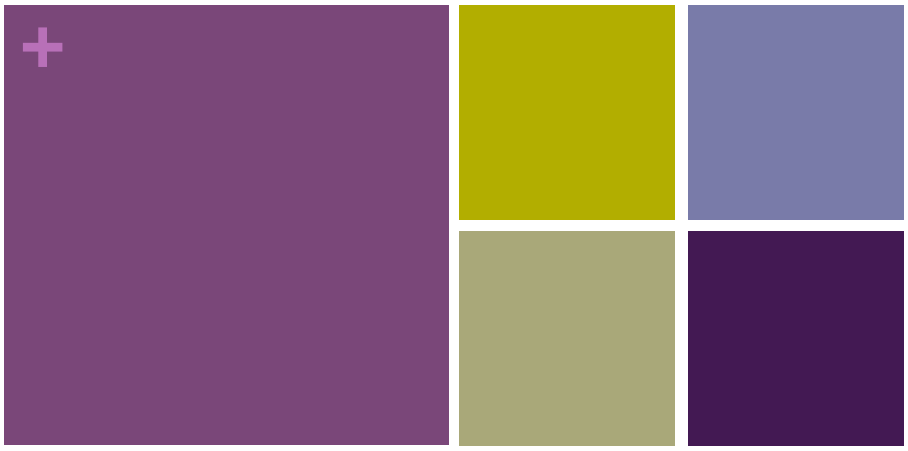


Control Structures and Collisions

Questions? / Announcements

- Assignment 3 due Next Monday Oct. 5.

Processing: Defining Functions

$$y = twice(x) = 2x$$

Return value | Function name | Function parameter | Function definition

```
float twice(float x) {
   return 2*x;
} // twice()
```


Example: Using random()

```
void setup() {// Create and set canvas
  size(300, 300);
  smooth();
  background(255);
} // setup()

void draw() {
  stroke(0);
  fill(random(255),
       random(255),
       random(255));
  ellipse(random(width),
  random(height),
  random(5, 20),
  random(5, 20));
} // draw();
```

Key Computing Ideas

- The computer follows a program's instructions. There are four modes:
  - **Sequencing**
    All statements are executed in sequence
  - **Function Application**
    Control transfers to the function when invoked
    Control returns to the statement following upon return
  - **Repetition**
    Enables repetitive execution of statement blocks
  - **Selection**
    Enables choice among a block of statements
- All computer algorithms/programs utilize these modes.

Function Application

- Control transfers to the function when invoked
- Control returns to the statement following upon return

```
void draw() {
  // Draw a barn at 50, 250 in 200 x (200 x 1.75) pixels
  barn(50, 250, 200, 200);
  barn(20, 100, 50, 50);
  barn(230, 100, 50, 75);
} // draw()
```

Parameter Transfer

50 250 200 200

```
void
barn(int barnX, int barnY, int wallWidth, int wallHeight) {
  // Draw a barn at <barnX, barnY> (bottom left corner)
  // with width wallWidth and height wallHeight * 1.75

 …

} // barn()
```

Repetition

- Enables repetitive execution of statement blocks

```
lather
rinse
repeat
```

```
/**
 * Repeat frameRate
 * times/second
 * Default frameRate = 60
 */
void draw() {
  lather(); // do this
  rinse(); // then this
  // and then this;
  // etc.
} // draw()
```

Loops: Controlled Repetition

- **While Loop**

```
while (<condition>) {
   stuff to repeat
}
```

- **Do-While Loop**

```
do {
   stuff to repeat
} while (<condition>)
```

- **For Loop**

```
for (<init>; <condition>; <update>) {
   stuff to repeat
}
```

All of these repeat the stuff in the block

The block {…} is called the Loop's Body

Writing Conditions in Processing

- Boolean expressions can be written using boolean operators.

Here are some simple expressions…

```
<          less than                 5 < 3
<=         less than/equal to        x <= y
==         equal to                  x == (y+j)
!=         not equal to              x != y
>          greater than              x > y
>=         greather than/equal to    x >= y
```

Logical Operations

- Combine two or more simple boolean expressions using logical operators:

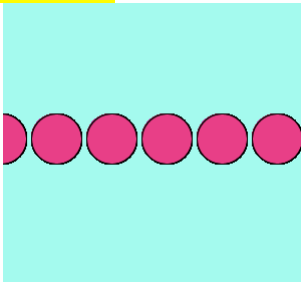
&&	and	(x < y) && (y < z)
||	or	(x < y) || (x < z)
!	not	! (x < y)

A	B	A && B	A || B	!A
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

Conditions in While Loops

```
while ( <condition> ) {
   stuff to repeat
}
```

```
int i = 0;
while (i < width) {
  ellipse(i, height/2, 50, 50);
  i = i + 55;
}
```



10,000 circles!

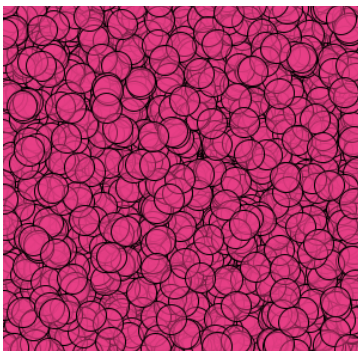
```
while ( <condition> ) {
   stuff to repeat
}
```

```
void setup() {
  size(300, 300);
  smooth();
  background(164, 250, 238);
  noLoop();
} // setup()

void draw() {

  fill(232, 63, 134, 127);
  stroke(0);

  int i = 0;
  while (i < 10000) {
    ellipse(random(width),
            random(height),
            25, 25);
    i = i + 1;
  }
} // draw()
```



Loops: Controlled Repetition

- **While Loop**

```
while (<condition>) {
    stuff to repeat
}
```

- **Do-While Loop**

```
do {
    stuff to repeat
} while (<condition>)
```

- **For Loop**

```
for (<init>; <condition>; <update>) {
    stuff to repeat
}
```

Do-While Loops

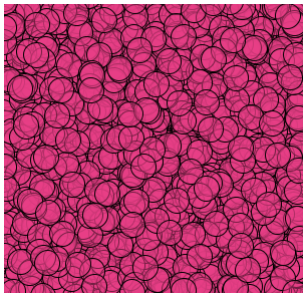
```
void setup() {
  size(300, 300);
  smooth();
  background(164, 250, 238);
  noLoop();
} // setup()

void draw() {

  fill(232, 63, 134, 127);
  stroke(0);

  int i = 0;
  do {
      ellipse(random(width),
              random(height),
              25, 25);
    i = i + 1;
  } while (i < 10000);
} // draw()
```

```
do {
    stuff to repeat
} while ( <condition> ) ;
```

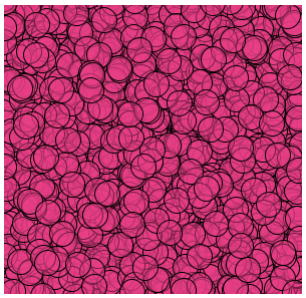


For Loops

```
for (<init>; <condition>; <update>)  {
   stuff to repeat
}
```

```
void setup() {
  size(300, 300);
  smooth();
  background(164, 250, 238);
  noLoop();
} // setup()

void draw() {
  fill(232, 63, 134, 127);
  stroke(0);
  for (int i = 0; i < 10000; i++) {
     ellipse(random(width),
             random(height),
             25, 25);
  }
} // draw()
```



Loops: Critical Components

- **Loop initialization**
  Things to do to set up the repetition
- **Loop Termination Condition**
  When to terminate the loop
- **Loop Body**
  The stuff to be repeated
- **Loop update**
  For the next repetition/iteration

Loops: Critical Components

Loop Initialization

```
for (int i = 0; i < 10000; i++) {
   ellipse(random(width),
           random(height),
           25, 25);
}
```

```
int i = 0;
while (i < 10000) {
  ellipse(random(width),
          random(height),
          25, 25);
  i = i + 1;
}
```

```
int i = 0;
do {
   ellipse(random(width),
           random(height),
           25, 25);
   i = i + 1;
} while (i < 10000);
```

Loops: Critical Components

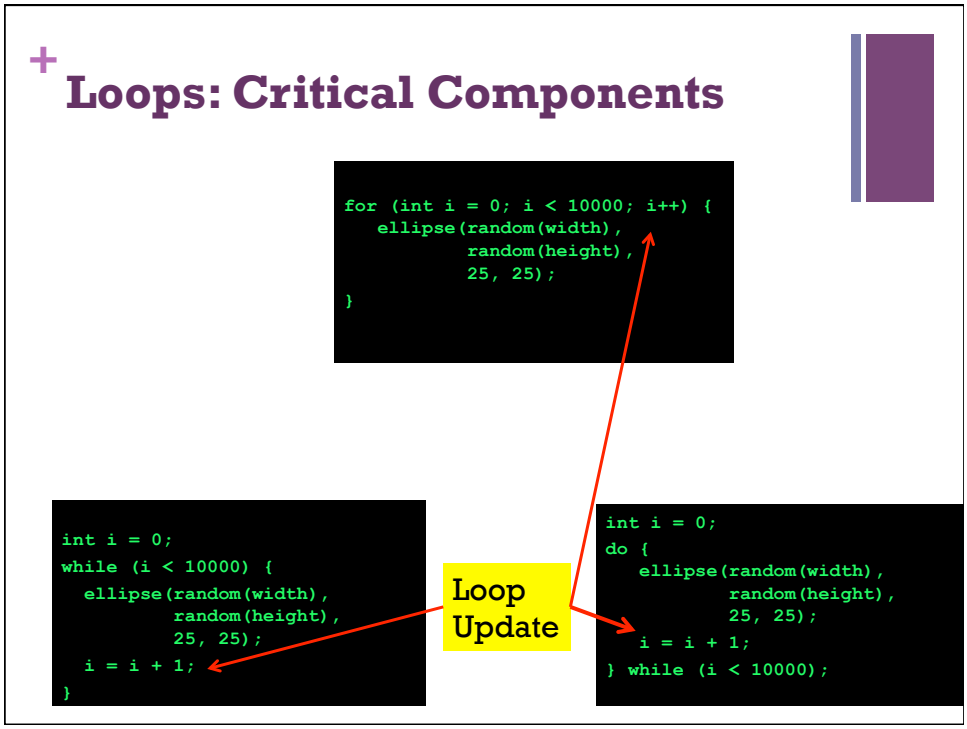
Termination Condition

```
for (int i = 0; i < 10000; i++) {
   ellipse(random(width),
           random(height),
           25, 25);
}
```

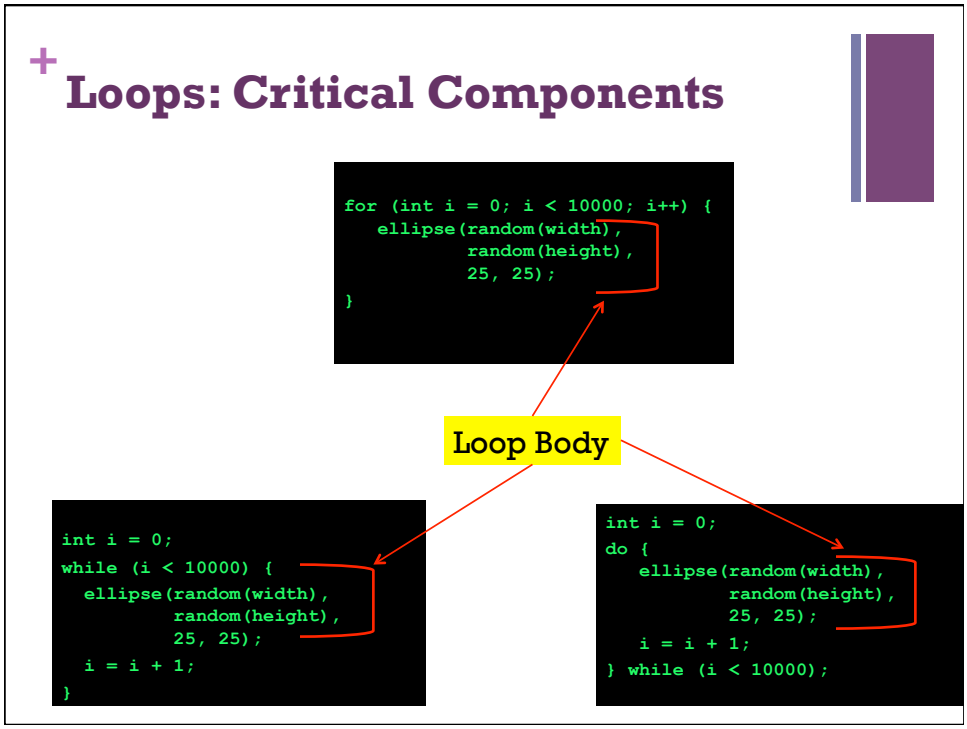
```
int i = 0;
while (i < 10000) {
  ellipse(random(width),
          random(height),
          25, 25);
  i = i + 1;
}
```

```
int i = 0;
do {
   ellipse(random(width),
           random(height),
           25, 25);
   i = i + 1;
} while (i < 10000);
```

Loops: Critical Components



Loops: Critical Components



Loops: Critical Components

- **Loop initialization**
  Things to do to set up the repetition
- **Loop Termination Condition**
  When to terminate the loop
- **Loop Body**
  The stuff to be repeated
- **Loop update**
  For the next repetition/iteration

What happens when any one of these is missing or incorrectly encoded??

Key Computing Ideas

- The computer follows a program's instructions. There are four modes:
  - **Sequencing**
    All statements are executed in sequence
  - **Function Application**
    Control transfers to the function when invoked
    Control returns to the statement following upon return
  - **Repetition**
    Enables repetitive execution of statement blocks
  - **Selection**
    Enables choice among a block of statements
- All computer algorithms/programs utilize these modes.

Selection

- Enables choice among a block of statements

Should I… { study }
{ sleep }
{ watch a movie }
{ veg out }
{ etc. }

- **If-statements** are one way of doing this

Selection: If Statement

```
if ( <condition> ) {
  do this
}
```

```
if ( <condition> ) {
  do this
}
else {
  do that
}
```

```
if ( <condition> ) {
  do this
}
else if ( <condition> ) {
  do that
}
else if (…) {
…
}
else {
  whatever it is you wanna do
}
```

At most ONE block is selected and executed.

Examples with if…

- Making things move
  - rotating rect examples
  - reference point example
  - for loop from a reference point.

Selection: Switch Statement

```
switch( <value> ) {
  case 'a':
  // do this if <value> == 'a'
  break;
  case 'b':
  // do this if <value> == 'b'
  break;
…
  default:
  // do this otherwise
  break;
```

At most ONE block is selected
(more than one block can be executed).

Selection: Switch Statement

```
switch( <value> ) {
  case 'a':
  // do this if <value> is 'a'
  case 'A':
  // do this if <value> is 'A' or 'a'
  break;
  case 'b':
  // do this if <value> is 'b'
  case 'B':
  // do this if <value> is 'B' or 'b'
  break;
…
  default:
  // do this otherwise
  break;
```

At most ONE block is selected
(more than one block can be executed).