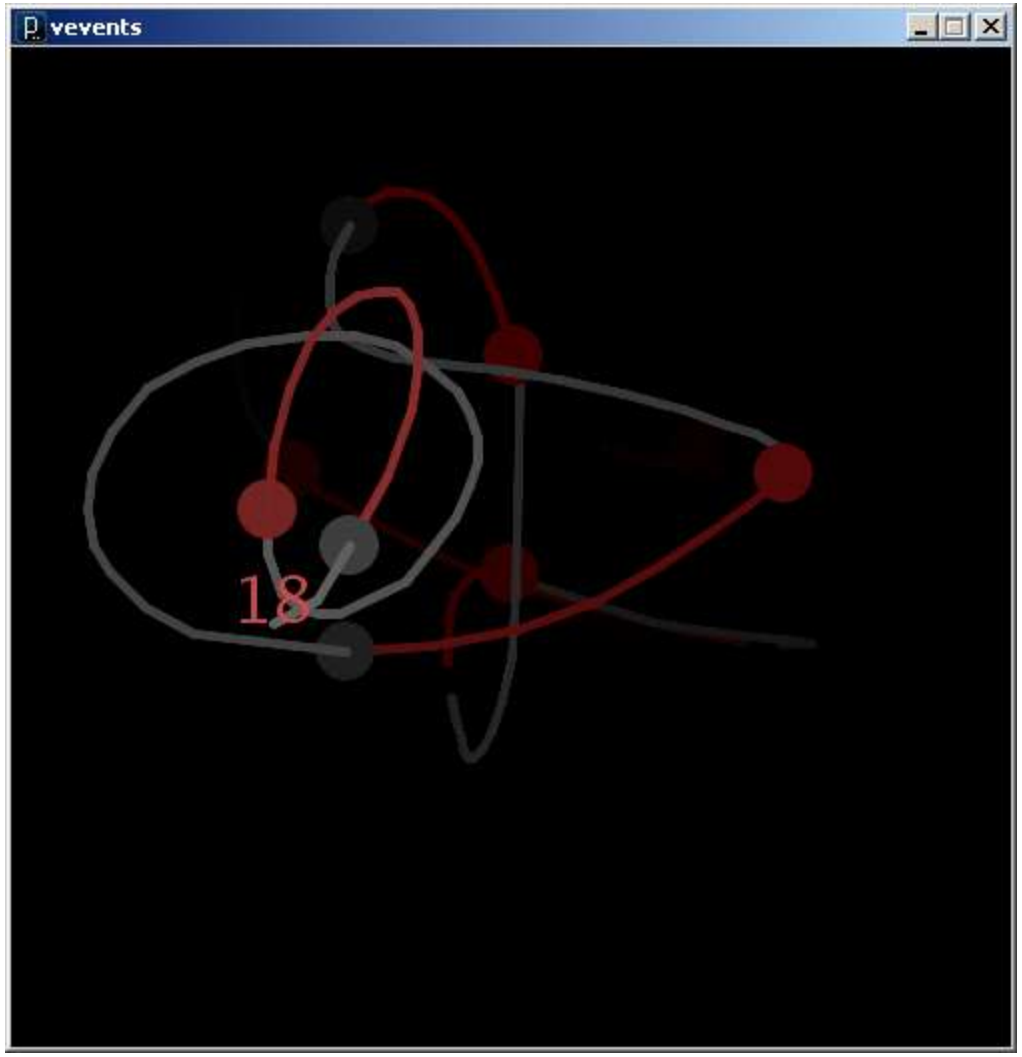
# Review

- Commenting your code
- Random numbers and printing messages
- mouseX, mouseY
- void setup() & void draw()
- frameRate(), loop(), noLoop()
- Arcs, curves, bézier curves, beginShape/endShape
- Example Sketches
- Dropbox
- Assignment #1

# vevents.pde



```
void mousePressed() {
   // Called when the mouse is pressed
}

void mouseReleased() {
   // Called when the mouse is released
}

void mouseClicked() {
   // Called when the mouse is pressed and released
   // at the same mouse position
}

void mouseMoved() {
   // Called while the mouse is being moved
   // with the mouse button released
}

void mouseDragged() {
   // Called while the mouse is being moved
   // with the mouse button pressed
}
```

```
void keyPressed() {
   // Called each time a key is pressed
}

void keyReleased() {
   // Called each time a key is released
}

void keyTyped() {
   // Called when an alpha-numeric key is pressed
   // Called repeatedly if the key is held down
}
```

## keyCode vs. key

key

- A built-in variable that holds the character that was just typed at the keyboard

keyCode

- A built-in variable that hold the numeric code for the keyboard key that was touched

All built-in keyboard interaction functions …

- set *keyCode* to the integer that codes for the keyboard key
- set *key* to the character typed
- All keyboard keys have a *keyCode* value
- Not all have a *key* value

# ASCII - American Standard Code for Information Interchange

| Char | Dec | Char | Dec | Char | Dec | Char | Dec | Char | Dec | Char | Dec | Char | Dec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (nul) | 0 | (dc4) | 20 | ( | 40 | < | 60 | P | 80 | d | 100 | x | 120 |
| (soh) | 1 | (nak) | 21 | ) | 41 | = | 61 | Q | 81 | e | 101 | y | 121 |
| (stx) | 2 | (syn) | 22 | * | 42 | > | 62 | R | 82 | f | 102 | z | 122 |
| (etx) | 3 | (etb) | 23 | + | 43 | ? | 63 | S | 83 | g | 103 | { | 123 |
| (eot) | 4 | (can) | 24 | , | 44 | @ | 64 | T | 84 | h | 104 | \| | 124 |
| (enq) | 5 | (em) | 25 | - | 45 | A | 65 | U | 85 | i | 105 | } | 125 |
| (ack) | 6 | (sub) | 26 | . | 46 | B | 66 | V | 86 | j | 106 | ~ | 126 |
| (bel) | 7 | (esc) | 27 | / | 47 | C | 67 | W | 87 | k | 107 | (del) | 127 |
| (bs) | 8 | (fs) | 28 | 0 | 48 | D | 68 | X | 88 | l | 108 | | |
| (ht) | 9 | (gs) | 29 | 1 | 49 | E | 69 | Y | 89 | m | 109 | | |
| (nl) | 10 | (rs) | 30 | 2 | 50 | F | 70 | Z | 90 | n | 110 | | |
| (vt) | 11 | (us) | 31 | 3 | 51 | G | 71 | [ | 91 | o | 111 | | |
| (np) | 12 | (sp) | 32 | 4 | 52 | H | 72 | \ | 92 | p | 112 | | |
| (cr) | 13 | ! | 33 | 5 | 53 | I | 73 | ] | 93 | q | 113 | | |
| (so) | 14 | " | 34 | 6 | 54 | J | 74 | ^ | 94 | r | 114 | | |
| (si) | 15 | # | 35 | 7 | 55 | K | 75 | _ | 95 | s | 115 | | |
| (dle) | 16 | $ | 36 | 8 | 56 | L | 76 | ` | 96 | t | 116 | | |
| (dc1) | 17 | % | 37 | 9 | 57 | M | 77 | a | 97 | u | 117 | | |
| (dc2) | 18 | & | 38 | : | 58 | N | 78 | b | 98 | v | 118 | | |
| (dc3) | 19 | ' | 39 | ; | 59 | O | 79 | c | 99 | w | 119 | | |

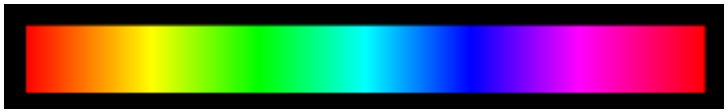
# More Color

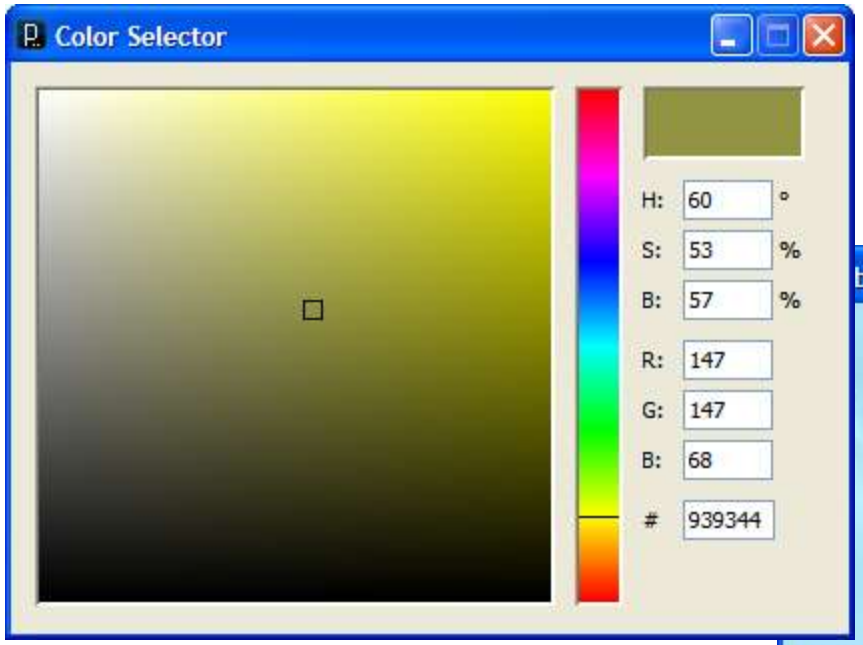
**`colorMode(`*`RGB or HSB`*`);`**

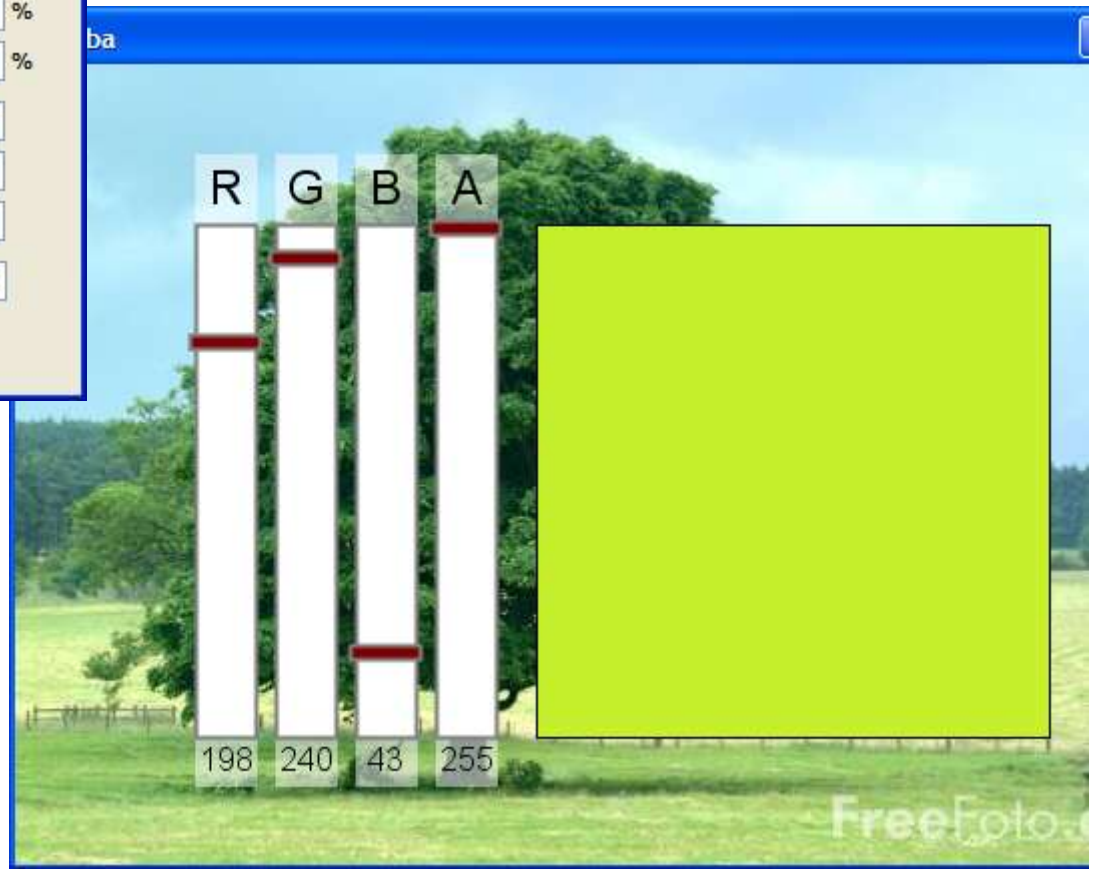
RGB: (red, green, blue)

HSB:

- hue
  - “pure color”
- saturation
  - “intensity”
- brightness
  - “lightness”




Color Selector
H: 60 °
S: 53 %
B: 57 %
R: 147
G: 147
B: 68
# 939344


ba
R
G
B
A
198
240
43
255

# Decimal vs. Binary vs. Hexadecimal

| Decimal | Hex | Binary |
|---|---|---|
| 0 | 00 | 00000000 |
| 1 | 01 | 00000001 |
| 2 | 02 | 00000010 |
| 3 | 03 | 00000011 |
| 4 | 04 | 00000100 |
| 5 | 05 | 00000101 |
| 6 | 06 | 00000110 |
| 7 | 07 | 00000111 |
| 8 | 08 | 00001000 |
| 9 | 09 | 00001001 |
| 10 | 0A | 00001010 |
| 11 | 0B | 00001011 |
| 12 | 0C | 00001100 |
| 13 | 0D | 00001101 |
| 14 | 0E | 00001110 |
| 15 | 0F | 00001111 |
| 16 | 10 | 00010000 |
| 17 | 11 | 00010001 |
| 18 | 12 | 00010010 |

counter.pde

# Primitive Data Types

| Type | Range | Default | Bytes |
|---|---|---|---|
| boolean | { true, false } | false | ? |
| byte | { 0..255 } | 0 | 1 |
| int | { -2,147,483,648 .. 2,147,483,647 } | 0 | 4 |
| long | { -9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807 } | 0 | 8 |
| float | { -3.40282347E+38 .. 3.40282347E+38 } | 0.0 | 4 |
| double | *much larger/smaller* | 0.0 | 8 |
| color | { #00000000 .. #FFFFFFFF } | *black* | 4 |
| char | *a single character* 'a', 'b', … | '\u0000' | 2 |

bits.pde / colorbits.pde

# Variables

- A *name* to which data can be assigned
- A variable is declared as a specific data type
- A variable is assigned a value using '='
- Variable names must begin with a letter, “_” or “$”
- Variables can container letters, digits, “_” and “$”

- Syntax: *type name;*
  *type name = expression;*

```
int i;
float x;
int j = 12;
boolean bReady = true;
float fSize = 10.0;
color _red = color(255,0,0);
```

*Variables are both declared and assigned a value*

# Rewrite randomEllipse using Variables

```
void draw() {
  fill( random(255), random(255), random(255) );
  ellipse(mouseX, mouseY, 30, 30);
}
```

---

```
void draw() {
  float R, G, B;
  R = random(255);
  G = random(255);
  B = random(255);
  fill( R, G, B );
  ellipse(mouseX, mouseY, 30, 30);
}
```

---

```
void draw() {
  float R = random(255);
  float G = random(255);
  float B = random(255);
  fill( R, G, B );
  ellipse(mouseX, mouseY, 30, 30);
}
```

# Using Variables

Draws a line from last mouse position to current.

Variables used to store last mouse position.

```
// Variables that store the last mouse pressed position.
int lastX;                          // Note where these are declared!
int lastY;

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

void draw() { /* must exist */ }

// Draw a line from the last mouse position
// to the current position.
void mousePressed() {
  line(lastX, lastY, mouseX, mouseY);
  lastX = mouseX;
  lastY = mouseY;
}
```

variables1.pde

# Using Variables

Orbit mouse with two shapes.

Variables used for temporary calculated values.

```
// Mouse orbiter
float angle;                    // Orbit angle state variable

void setup() {
  size(500, 300);
  background(255);
}

void draw() {
  background(255);
  fill(0, 0, 255);
  angle = angle + 0.3;          // Increment angle
  float dX = 30.0*cos(angle);   // Mouse position offset
  float dY = 30.0*sin(angle);   // Draw two orbiting shapes
  ellipse(mouseX + dX, mouseY + dY, 5, 5);
  ellipse(mouseX - dX, mouseY - dY, 5, 5);
}
```

variables2.pde

# Data Type Conversion

- Variables of some types can be converted to other types.
- Type conversion function names are the types to which data will be converted.

```
// binary(…), boolean(…), byte(…),
// char(…), float(…), str(…)

float f = 10.5;
int i;

//i = f;                  // Throws a runtime error
i = int(f);

println( char(65) );    // Prints the character 'A'
```

# Other "things" …

| Type | Range | Default | Bytes |
|---|---|---|---|
| String | a series of chars in quotes "abc" | null | ? |
| PImage | an image | null | ? |
| PFont | a font for rendering text | null | ? |
| … | | | |

```
String message = "Hello World!";
```

# Images

**`PImage img;`**

- Declares a variable to hold an image

**`img = loadImage(`*`filename`*`);`**

- Loads an image from a file in the *data* folder in sketch folder.
- Must be assigned to a variable of type PImage.

**`image(`*`img, X, Y, [X2, Y2]`*`);`**

- Draws the image *img* on the canvas at X, Y
- Optionally fits image into box X,Y and X2,Y2

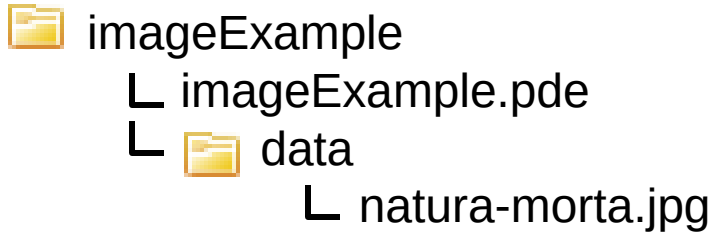
**`imageMode(CORNER);`**

- X2 and Y2 define width and height.

**`imageMode(CORNERS);`**

- X2 and Y2 define opposite corner.

# Image Example




```
PImage img;

void setup()
{
  size(500, 400);
  img = loadImage("natura-morta.jpg");
  image(img, 50, 40);
}
```

imageExample.pde

# Expressions

- Series of data values, variables and/or sub-expressions, related by operators and functions, and grouped by parentheses.

- Expressions are automatically evaluated and replaced by the final evaluated value.

- Expressions can be assigned to variables using "="
  - Expression is always on right
  - Variable name is always on left

*variable_name* **=** *expression;*

# Operators

Symbols that operate on one or two sub-expressions.

Infix, prefix, or postfix

- Mathematical ( `+, -, *, /, …` )
  - Perform standard mathematical operations (PEMDAS)
- Relational ( `<, >, ==, !=, …` )
  - Test relationship between related expressions.
  - Always returns a boolean value (true or false).
- Logical ( `&&, ||, !` )
  - Logical conjunction (and), disjunction (or), negation (not).
  - Always returns a boolean value (true or false).

# Mathematical Operators

| | |
|---|---|
| `+, -, *, /` | and … |
| `i ++;` | *equivalent to* `i = i + 1;` |
| `i += 2;` | *equivalent to* `i = i + 2;` |
| `i --;` | *equivalent to* `i = i - 1;` |
| `i -= 3;` | *equivalent to* `i = i - 3;` |
| `i *= 2;` | *equivalent to* `i = i * 2;` |
| `i /= 4;` | *equivalent to* `i = i / 4;` |
| `i % 3;` | the remainder after i is divided by 3 (modulo) |

Examples:

```
1 + 2
slope = (y2 - y1) / (x2 - x1);
i++
```

# Relational Operators

| | |
|---|---|
| < | less than |
| > | is greater than |
| <= | is less than or equal to |
| >= | is greater than or equal to |
| == | is equivalent |
| != | is not equivalent |

Examples:

```
true
10 >= 10
'A' != 'A'
```

# Logical Operators

`&&` logical conjunction (and)

both expressions must evaluate to 'true' for conjunction to evaluate to 'true'

`||` logical disjunction (or)

either expression must evaluate to 'true' for disjunction to evaluate to 'true'

`!` logical negation (not)

!true $\rightarrow$ false, !false $\rightarrow$ true

Examples:

```
true && true
true || false
!false
```

# Evaluating Logical Expressions

Negation ( !A )

| A | !A |
|---|---|
| false | true |
| true | false |

Conjunction "AND" (A && B)

| A | B | A && B |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | false |

Disjunction "OR" (A || B)

| A | B | A \|\| B |
|---|---|---|
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

Derive new tables by combining operators…

1. *If I've already had two desserts, then don't serve me any more. Otherwise, I'll take another, thank you.*

   *A = had_dessert_1, B = had_dessert_2*

!(A && B) or !A || !B

| A | B | !(A && B) |
|---|---|---|
| true | true | false |
| true | false | true |
| false | true | true |
| false | false | true |

2. *I'll have dessert, as long as it is not flan (A) or beef jerky (B).*

# Some Built-in Mathematical Functions

```
sin(x), cos(x), tan(x), asin(x), …
abs(x), exp(x), pow(x, y), log(x), sqrt(x), …
max(x1, x2), min(x1, x2), floor(x), ceil(x), …
```

`dist(x1, y1, x2, y2)` -> distance between two points

`norm(value, low, high)` -> normalizes a value to [0-1]

… and many more, all of which can be included in an expression.

# Evaluating Expressions

```
1 + 2
pow(sin(x),2) + pow(cos(x),2) == 1.0
max(1, 2, 3) >= 2
floor(2.9) == ceil(1.8)


void setup()
{
    float r = 200.0;
    size( 500, 200+300 );
    background( 0.5*r, 0, 0 );
}
```