Review

- `setup()` & `draw()`
- The event loop
- `mouseX`, `mouseY`
- Mouse and Keyboard interactions
- Arcs, curves, bézier curves, custom shapes
- Programming principals
  - Syntax is important
  - Reference manuals are your friend
  - Don't be afraid to try different things

Mouse Interaction

- Built-in predefined variables that hold the mouse X and Y locations
  - current `mouseX` `mouseY`
  - previous (last) `pmouseX` `pmouseY`
  - 0 if mouse is not in window
- Built-in predefined variables that indicate the button state:
  - is the `mousePressed`?
  - which `mouseButton`?
    - LEFT
    - RIGHT
    - CENTER

```java
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
}

void keyPressed() {
  // Called each time a key is pressed
}
void keyTyped() {
  // A variable: true when a key is currently being pressed
  keyTyped(); // a variable: true when a key is currently being pressed
}
void keyReleased() {
  // Called each time a key is released
}

void keyPressed() {
  // Called each time a key is pressed
  keyPressed();
}
void keyTyped() {
  // Called when a key is pressed
  keyTyped();
}
void keyReleased() {
  // Called when a key is released
  keyReleased();
}
```

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 holds the 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

<table>
<thead>
<tr>
<th>ASCII - American Standard Code for Information Interchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f</td>
</tr>
<tr>
<td>00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f</td>
</tr>
</tbody>
</table>
Text

text(theString, x, y);
  − Draws theString on the sketch at (x, y)
  − A string is represented by ""
  − text("CS110 is fun!", width/2, height/2);

textSize(size);
  − Sets the current font size

randomEllipsoid

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

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

Variables

• A location where data is stored
• A variable name is declared as a specific data type
• Names must begin with a letter, "_" or "$" and can contain letters, digits, "_" and "$"

boolean isTuesday = true;
int i;
int j = 12;
float fSize = 10.0;
color _red = color(255,0,0);
String name123 = "Fred";
PImage img;

Variable Uses

• Refer to a value throughout your program
  − but allow it to be changed
  − As temporary storage for a intermediate computed result
  − To parameterize – instead of hardcoding coordinates
• Special variables (preset variables)
  − width, height
  − mouseX, mouseY, pmouseX, pmouseY
• Assigned with a single =
  − known as the assignment operator
  − left side and right side are not equal

Primitive Data Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>{true, false}</td>
<td>false</td>
<td>?</td>
</tr>
<tr>
<td>byte</td>
<td>{0..255}</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>int</td>
<td>{-2,147,483,648..2,147,483,647}</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>long</td>
<td>{-9,223,372,036,854,775,808..9,223,372,036,854,775,807}</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>float</td>
<td>{-3.40282347E+38..3.40282347E+38}</td>
<td>0.0</td>
<td>4</td>
</tr>
<tr>
<td>double</td>
<td>much larger/smaller</td>
<td>0.0</td>
<td>8</td>
</tr>
<tr>
<td>color</td>
<td>{#0000000..#FFFFFF}</td>
<td>black</td>
<td>4</td>
</tr>
<tr>
<td>char</td>
<td>a single character 'a', 'b',...</td>
<td>'u0000'</td>
<td>2</td>
</tr>
</tbody>
</table>
Other "things" ...

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>a series in quotes &quot;abc&quot;</td>
<td>null</td>
<td>?</td>
</tr>
<tr>
<td>PImage</td>
<td>an image</td>
<td>null</td>
<td>?</td>
</tr>
<tr>
<td>PFont</td>
<td>a font for rendering text</td>
<td>null</td>
<td>?</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Data Type Conversion

- Types must match
- If variable types on the two sides of an assignment do not match, one must be converted
  - automatic conversion
  - explicit conversion (casting)

```
float f = 10.0;
int i = 5;

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

Images

```
save(filename);
loadImage(filename);

- Loads an image from a file in the sketch folder.
- Or in the data subfolder.
- 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 (resize)

imageMode(CORNER):
- X and Y define the upper left corner
- X2 and Y2 define width and height.
```

Image Example

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

Conditionals: if-statement

```
if (boolean_expression) {
  statements;
}
```

What does this do?

```
void draw() {
  if (mouseX > 50 && mouseY > 50) {
    ellipse( mouseX, mouseY, 10, 10 );
  }
}
```
Logical Expressions

&& logical conjunction (and)
  • both expressions must be true for conjunction to be true

|| logical disjunction (or)
  • either expression must be true for disjunction to be true

! logical negation (not)
  • true → false, false → true

Relational Expressions

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

Relational Expressions: Examples

1. if (true) { .. }
2. if (10 > 10) { .. }
3. if (10 >= 10) { .. }
4. if ('a' == 'a') { .. }
5. if ('a' != 'a') { .. }
6. if ("Bryn Mawr" != "bryn mawr") { .. }

Logical Expression Examples

1. if ((2 > 1) && (3 > 4)) { .. }
2. if ("blah" == "blah") && (1 + 2 == 3) { .. }
3. if (!false) { .. }
4. if (!(-1 < 1)) { .. }
5. if (!((10 < 20) || false)) { .. }
6. if (!((10 > 20) && (10 < 20))) { .. }
7. if ((true || false) && true) { .. }
8. if ((true && false) || true) { .. }
9. ..

Conditionals: if-else-statement

```java
if ( boolean_expression ) {
    statements executed when boolean_expression is true;
} else {
    statements executed when boolean_expression is false;
}
```

What does this do?

```java
void draw() {
    if (mouseY < 50) {
        println("the sky");
    } else {
        println("the ground");
    }
}
```

Conditionals: if-else-if-statement

```java
if ( boolean_expression_1 ) {
    statements;
} else if ( boolean_expression_2 ) {
    statements;
} else if ( boolean_expression_3 ) {
    statements;
} else {
    statements;
}
```
What does this do?

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

void draw() {
    if (mouseX < width/2) {
        if (mouseY < height/2) {
            fill(0, 255, 0);
        }
        else {
            fill(0, 0, 255);
        }
    }
    else {
        if (mouseY < height/2) {
            fill(255, 0, 0);
        }
        else {
            fill(255);
        }
    }
    ellipse(mouseX, mouseY, 50, 30);
}
```

And this?

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

void draw() {
    if (mouseX > 100) {
        background(255, 0, 0);
    }
    else if (mouseX > 200) {
        background(0, 0, 255);
    }
}
```

Does this work better?

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

void draw() {
    if (mouseX > 200) {
        background(255, 0, 0);
    }
    if (mouseX > 100) {
        background(0, 0, 255);
    }
}
```

Simulated Motion (balldrop)

- \( p \) = position
- \( v \) = velocity
- \( a \) = acceleration

- Constant acceleration \( a \)
  - assuming small time intervals \( t=1 \)
  \[
  p_{n+1} = p_n + vt \\
  v_{n+1} = v_n + a
  \]