Basic Input and Output
CSE 120 Spring 2017

Instructor: Justin Hsia
Teaching Assistants: Anupam Gupta, Braydon Hall, Eugene Oh, Savanna Yee
Administrivia

- Assignments:
  - Animal Functions due today (4/12)
  - Reading Check 3 due tomorrow *before lab* (4/13)
  - Jumping Monster due Saturday (4/15)

- “Big Idea” this week: Algorithms
Outline

- Other Useful Processing Tools
- User Input and Output
  - Mouse (input)
  - Keyboard (input)
  - Text (output)
System Variables

- Special variables that hold values related to the state of the program, often related to user input
  - You don’t need to declare these variables
  - These variables will update automatically as the program runs
  - Colored pink/magenta-ish in the Processing environment

- We’ve used some of these already:
  - `mouseX, mouseY, width, height`

- We’ll see more today
Transparency/Opacity

- You can add a 4\textsuperscript{th} argument to a color!
  - This also applies to the \texttt{fill()} and \texttt{stroke()} functions
- This argument also takes an integer between 0–255
  - 0 is fully transparent (invisible)
  - 255 is fully opaque (the default)

```javascript
size(400, 320);
noStroke();
background(136, 177, 245);

fill(255, 0, 0, 100);
ellipse(132, 120, 200, 200);
fill(0, 200, 0, 150);
ellipse(200, 200, 200, 200);
fill(0, 0, 200, 50);
ellipse(268, 118, 200, 200);
```
Custom Shapes

- Define vertices between `beginShape()` and `endShape()`
  - If planning to reuse, best to create in a separate function

```plaintext
size(480, 240);
fill(153, 176, 180);
beginshape();
  vertex(100, 240);
  vertex(200, 180);
  vertex(220, 120);
  vertex(160, 40);
  vertex(420, 120);
  vertex(320, 160);
  vertex(400, 180);
  vertex(280, 200);
  vertex(260, 240);
endshape();
fill(0);
ellipse(310, 120, 16, 16);
```
Drawing and Frames

- Control and track how frequently `draw()` runs
  - Each time `draw()` runs, it is called a new *frame*

- `frameRate()` changes the desired number of frame updates per second
  - So larger argument is faster
  - Default is `frameRate(60)`

- System variable `frameCount` returns the number of frames since the start of the program
  - Starts at 0 in `setup()`
Drawing and Frames

- Control and track how frequently `draw()` runs
  - Each time `draw()` runs, it is called a new `frame`

- `noLoop()` stops `draw()` from being continuously executed
  - Can restart using `loop()`
Outline

- Other Useful Processing Tricks
- **User Input and Output** *
  - Mouse
  - Keyboard
  - Text

* We will look at a subset of the available Processing commands. For a full list, see the Processing Reference.
The Mouse

- System variables:
  - `mouseX` – x-coordinate of mouse in current frame
  - `mouseY` – y-coordinate of mouse in current frame
  - `pmouseX` – x-coordinate of mouse in previous frame
  - `pmouseY` – y-coordinate of mouse in previous frame
  - `mousePressed` – is a button currently being pressed?

- Built-in functions:
  - `mousePressed()` – called very time a button is pressed
  - `mouseReleased()` – called every time a button is released
Example: Path Drawing

- Last lecture we wrote a *dot*-drawing program

- We can additionally use `pmouseX` and `pmouseY` to create a *path*-drawing program

```java
void setup() {
  size(500, 500);  // set drawing canvas size
  strokeWeight(8);  // thicker lines
  stroke(0, 0, 0, 120);  // black line with some transparency
  frameRate(30);  // slow down the frame rate
}

void draw() {
  line(mouseX, mouseY, pmouseX, pmouseY);
}
```
Example: Frame Dots

- Slow down to 1 frame per second and have a new dot appear on each frame
  - Number of dots on screen will equal current frame count
  - Reminder: `frameRate()`, `frameCount`
  - Calculate position using division and modulus

- Control the animation with the mouse
  - Pause while mouse is pressed
  - Reminder: `loop()`, `noLoop()`
Hovering Over a Rectangle

if (mouseX >= x)

if (mouseX <= x + w)
Hovering Over a Rectangle

if (mouseY >= y)

if (mouseY <= y + h)
Hovering Over a Rectangle

```
if ( (mouseX >= x)    &&
    (mouseX <= x + w) &&
    (mouseY >= y)     &&
    (mouseY <= y + h) )
```
Hovering Over a Rectangle

```cpp
int x = 100; // x-position of upper-left corner
int y = 160; // y-position of upper-left corner
int w = 200; // width of rectangle
int h = 160; // height of rectangle

void setup() {
    size(500,500); // set drawing canvas size
    noStroke(); // no shape outlines
}

void draw() {
    background(204); // clear the canvas

    if ((mouseX >= x) && (mouseX <= x+w) && (mouseY >= y) && (mouseY <= y+h)) {
        fill(0); // black is mouse is hovering over
    } else {
        fill(255); // white otherwise
    }

    rect(x, y, w, h); // draw the rectangle
```
The Keyboard

- **System variables:**
  - `key` – stores the ASCII value of the last key press
  - `keyCode` – stores codes for non-ASCII keys (e.g. UP, LEFT)
  - `keyPressed` – is any key currently being pressed?

- **Built-in functions:**
  - `keyPressed()` – called every time a key is pressed

- **New datatype:** `char`
  - Stores a single character (really just a number)
  - Should be surrounded by *single* quotes
  - *e.g.* `char letter = 'a';`
Example: Keyboard Dots

```cpp
int position = 0;

void setup() {
    size(400, 100);
    noStroke();
    background(0);
    fill(0);
}

void draw() {
    ellipse(position, 40, 40, 40);
}

void keyPressed() {
    if(key == 'g') {
        fill(0, 255, 0);
    }
    if(key == 'y') {
        fill(255, 255, 0);
    }
    if(key == 'm') {
        fill(255, 0, 255);
    }
    position = position + 50; // position+=50;
}
```
Text Output

- `println(yourText);`
  - Prints `yourText` to the console, which is the black area below your Processing code
  - Useful for debugging, particularly your portfolio

- `text(yourText, x, y);`
  - Prints `yourText` on the drawing canvas, starting with the bottom-left corner at coordinate `(x, y)`
  - Change the size of your text using `textSize(size);`

- `yourText` should be between `double` quotes
  - We will talk about the datatype `String` later
Example: Displaying Typed Keys

```java
void setup() {
    size(120, 120);
    textSize(64);
    textAlign(CENTER);
}

void draw() {
    background(0);
    text(key, 60, 80);
}
```
Example: Moving a Rectangle

- Reminder: arrow keys (UP, DOWN, LEFT, RIGHT) are coded keys

```java
if (keyPressed) {
    if (key == CODED) {
        if (keyCode == LEFT) {
            x = x - 1;
        }
    }
}
```
Example: Moving a Rectangle

```cpp
int x = 215;

void setup() {
    size(480, 120);
}

void draw() {
    background(0);
    rect(x, 45, 50, 50);

    if(keyPressed) {
        if(key == CODED) {
            if(keyCode == LEFT) {
                x = x - 1;
            }
            if(keyCode == RIGHT) {
                x = x + 1;
            }
        }
    }
}
```
Looking Forward

- **Next week is the Creativity Assignment**
  - In pairs, you will be asked to create and submit TWO Processing projects *of your choice*
  - The point is to use the tools available to you to make something fun and creative!
  - Planning document due Tuesday (4/18)
  - Actual programs due next, next Monday (4/24)

- **Portfolio Update 1** is due Tuesday (4/18)
  - Building a Robot, Logo Design, Lego Family, Animal Functions, Jumping Monster
  - Ask your TAs for assistance!