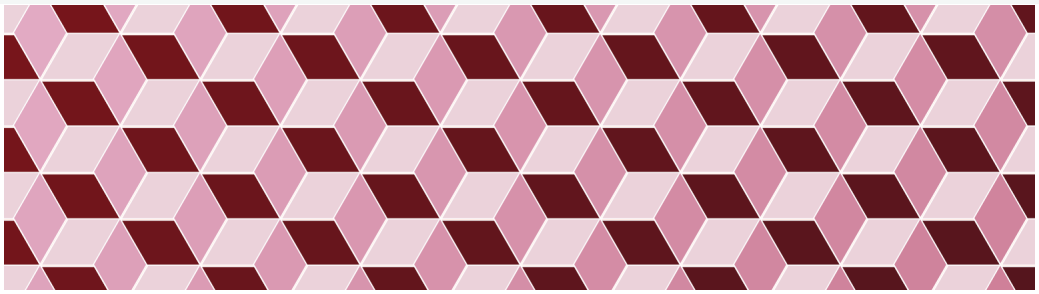


15-104 Introduction to Computing for Creative Practice

Fall 2022

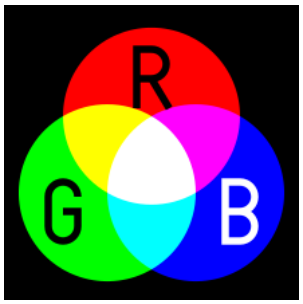
02 Basics of p5.js Programming (cont'd)

Instructor: Tom Cortina, tcortina@cs.cmu.edu, GHC 4117, 412-268-3514

1

RGB

- The RGB color model is *additive* in the sense that the three light beams are added together, and their light spectra add, wavelength for wavelength, to make the final color's spectrum. (Wikipedia)



- https://www.rapidtables.com/web/color/RGB_Color.html

2

Color!

- Pixels can have color (of course). A popular model for programming is the **RGB model**, where each pixel is made up of a mixture of red, green and blue. (we will use others later)
- `fill(r, g, b, [alpha]);`
  `stroke(r, g, b, [alpha]);`
  Parameters:

`r`	Number: Amount of red (0 to 255, inclusive).
`g`	Number: Amount of green (0 to 255, inclusive).
`b`	Number: Amount of blue (0 to 255, inclusive).
`alpha`	Number: Opacity (0 = transparent, to 255=fully opaque).

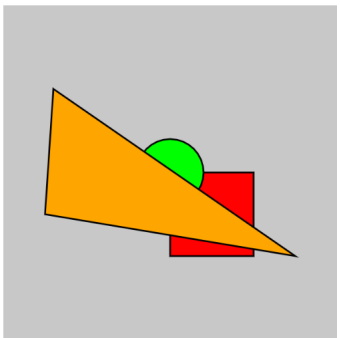
3

Drawing is like painting…

- It's sequential. New paint goes on top of old paint.
  - The order you write the instructions is the order that your painting will be constructed.

```
function draw() {
  background(200);
  fill(255, 0, 0);          // red
  rect(100, 100, 50, 50);
  fill(0, 255, 0);          // green
  circle(100, 100, 40);
  fill(255, 165, 0);        // orange
  triangle(30, 50, 25, 125, 175, 150);
}
```

Comment for ourselves

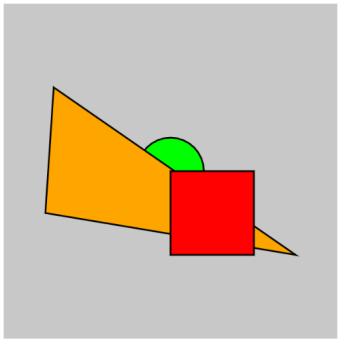


4

Drawing is like painting…

- Same set of commands, why does this painting look different?

```
function draw() {
  background(200);
  fill(0, 255, 0);          // green
  circle(100, 100, 40);
  fill(255, 165, 0);        // orange
  triangle(30, 50, 25, 125, 175, 150);
  fill(255, 0, 0);          // red
  rect(100, 100, 50, 50);
}
```

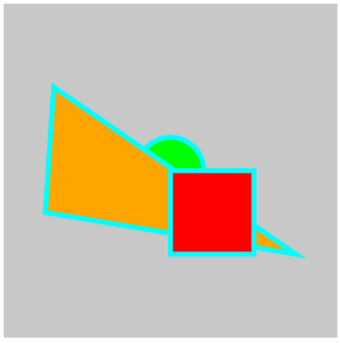


5

What's going on?

In this example, we want to draw only the red rectangle with a cyan border, but all the shapes get a cyan border! Why?

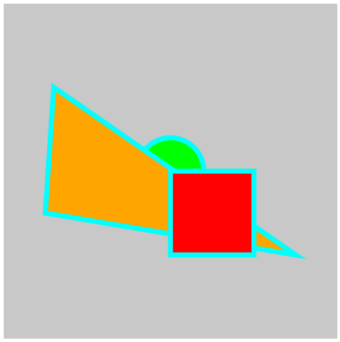
```
function draw() {
  background(200);
  fill(0, 255, 0);          // green
  circle(100, 100, 40);
  fill(255, 165, 0);        // orange
  triangle(30, 50, 25, 125, 175, 150);
  stroke(0, 255, 255);      // cyan
  strokeWeight(3);
  fill(255, 0, 0);          // red
  rect(100, 100, 50, 50);
}
```



6

The `draw` function loops!

- Remember that the `draw` function automatically loops/repeats itself.
- So once we change the stroke color and weight, that color and weight remain in effect for the next repetition of `draw`.
- If you are drawing a painting (static image) and don't want the draw function to repeat itself, you can call the `noLoop()` function at the end of the `draw` function so it doesn't loop back on itself.

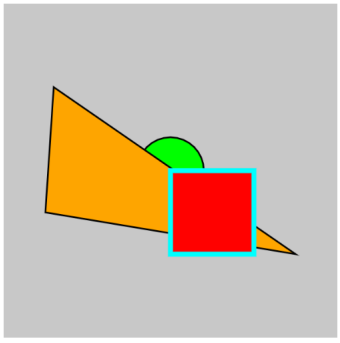


7

noLoop()

That's better!

```
function draw() {
  background(200);
  fill(0, 255, 0);        // green
  circle(100, 100, 40);
  fill(255, 165, 0);      // orange
  triangle(30, 50, 25, 125, 175, 150);
  stroke(0, 255, 255);    // cyan
  strokeWeight(3);
  fill(255, 0, 0);        // red
  rect(100, 100, 50, 50);
  noLoop();
}
```



8

Mouse input (introduction)

`mouseX`
`mouseY`

- The system variable `mouseX` always contains the current horizontal position of the mouse, relative to (0, 0) of the canvas.
- The system variable `mouseY` always contains the current vertical position of the mouse, relative to (0, 0) of the canvas.

```
function draw() {
    background(0);
    fill(0, 230, 130);
    ellipse(mouseX, mouseY, 140, 95);
}
```

9

Random values (introduction)

`random(x, y)`

- Returns a random floating-point between `x` (inclusive) and `y` (exclusive).

```
function draw() {
    background(200);
    fill(250, 250, 0);
    ellipse(random(0, 300), random(0, 300), random(10, 140), random(9, 120));
    ellipse(random(0, 300), random(0, 300), random(10, 140), random(9, 120));
    ellipse(random(0, 300), random(0, 300), random(10, 140), random(9, 120));
    noLoop();
}
```

10

Code Style: Comments

- Comments help explain parts of your code to the reader

```
fill (255, 255, 255);  // this is a comment (to the end of the line)
// this is a comment too
/* this is also a comment between the slash-star and star-slash */
/* this is a comment
   over several lines */
```

- Comments help us understand our code months from now when we go back to it to update it or use part of it for another program.

11

Code Style: Indentation

- Shows code that is "inside" other code.
- Example: the body of a function is inside its function declaration:

```
function draw() {
    background(200);
    ellipse(50, 50, 80, 80);
}
```

- Typically, a left bracket { increases indentation and a right bracket } decreases it.
- Use 4 spaces. Never use tabs.
- See the website if you use Sublime to make sure tabs translate to 4 spaces.

12

No!

- This will still work, but you will make us sad, or mad.

```
function draw() {
background(200);
fill(0, 255, 0);
circle(100, 100, 40);
fill(255, 165, 0);
triangle(30, 50, 25, 125, 175, 150);
fill(255, 0, 0);
rect(100, 100, 50, 50);
}
```

13

Console

Google Chrome
View
> Developer
>> Javascript Console

- Look for the Javascript console in your browser.
- This will help you find errors in your code.

Example: `Ellipse(mouseX, mouseY, 140, 95);`

Console message:

```
Uncaught ReferenceError: Ellipse is not defined
    at draw (sketch.js:10)
    ...
```

The error indicates that it can't recognize `Ellipse` (should be `ellipse`) in the `draw` function in `sketch.js` at line number 10.

14

Using Arithmetic

- In general, at any place you can write a number, you can write an arithmetic expression or a function call that evaluates to a number.
- Example:

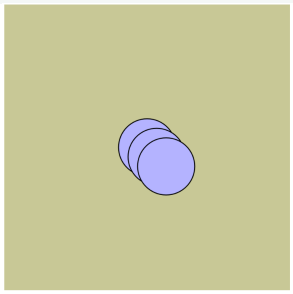
```
ellipse(100, 100, 50, 75);

ellipse(mouseX / 2, mouseY / 2, 50, 75);

ellipse(100, 100, random(30, 70), random(40, 60));
```

15

Example



```
function setup() {
    createCanvas(300, 300);
}

function draw() {
    background(200, 200, 150);
    fill(180, 180, 255);
    ellipse((width / 2), (height / 2), 60, 60);
    ellipse((width / 2) + 10, (height / 2) + 10, 60, 60);
    ellipse((width / 2) + 20, (height / 2) + 20, 60, 60);
}
```

`width` is a p5.js environment variable that represents the width of the canvas in pixels.
`height` is a p5.js environment variable that represents the height of the canvas in pixels.
Why are these identifiers useful?

16

Conditionals (the `if` statement)

- An `if` statement allows to test a logical condition to determine whether to run some code or not.
- Logical conditions are expressions that evaluate to true or false.
- Expressions with the relational operators lead to true or false:

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

17

Example

```
function setup() {
    createCanvas(300, 300);
}


function draw() {
    background(230, 230, 0);
    if (mouseX < (width / 2)) {
        background(0, 0, 200);
    }
}
```

Note the parentheses around the logical condition:

```
if ( condition ) {
    // body of if statement
}
```

Note the instruction(s) that are to be executed only if the condition is true are inside the brackets and indented.

18

Conditionals (the `if/else` statement)

- An `if-else` statement allows to test a logical condition to determine whether to run some code or some other code.
- General forms for `if` and `if-else`:

```
if ( condition ) {
    instruction(s) if true
}
```

```
if ( condition ) {
    instruction(s) if true
} else {
    instruction(s) if false
}
```

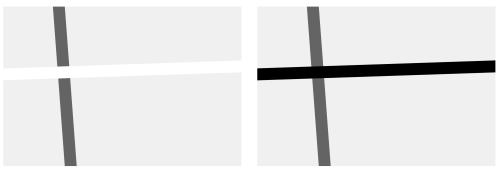
19

Example (McCarthy, Reas, Fry)

`mouseIsPressed` is a p5.js system variable that evaluates to true if the mouse is pressed down and false otherwise.

```
function setup() {
    createCanvas(600, 400);
    strokeWeight(30);
}
```

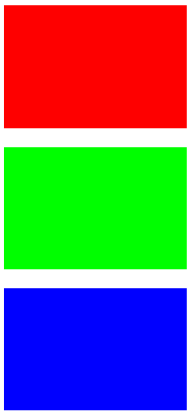
```
function draw() {
    background(240);
    stroke(102);
    line(140, 0, 170, height);
    if (mouseIsPressed) {
        stroke(0);
    } else {
        stroke(255);
    }
    line(0, 170, width, 150);
}
```



20

Example (layered `if`s)

```
function setup() {
    createCanvas(600, 400);
}
function draw() {
    if (mouseX < (width * 0.33)) {
        background(255, 0, 0);
    } else if (mouseX > (width * 0.66)) {
        background(0, 255, 0);
    } else {
        background(0, 0, 255);
    }
}
```



21

Try these:

- Put a 20 X 20 square in each corner of a canvas (assuming the canvas is at least 40 X 40).
- Make a circle appear if the mouse is below the middle of your canvas.
  - Modify the code so the circle follows the mouse if the mouse is below the middle of the canvas, but nothing appears otherwise.
- Make the background of the canvas turn red if the mouse is in the top left quadrant, green if the mouse is in the bottom left quadrant, blue if the mouse is in the top right quadrant, and black if the mouse is in the bottom right quadrant. (HINT: You can do this by layering `if-else` instructions.)

22