

Code-Moji

By: Cassandra Lenters

Duration: 120 minutes

LEVEL	SUBJECTS	PROVINCES / TERRITORIES	TOOL
Grades 4-6, 7-8	Art, Mathematics	Across Canada	Processing, OpenProcessing

Overview

Create emojis with code! In this activity, students remix a Processing sketch to build their very own interactive emoji artwork.

Prep Work

- Create a (free) account for your class at <https://www.openprocessing.org/home/join>
- Review the example project: <https://www.openprocessing.org/sketch/559871>
- Print the solution sheet: <http://bit.ly/codemoji-solution> (see below)
- This lesson requires computers and access to the internet

Lesson

Introduction

Watch this video about the History of Emoji:
<http://bit.ly/emoji-history> (~5 minutes)

Ask: "What stood out to you?" "What is something you

Key Coding Concepts

- ✓ Algorithms
- ✓ Conditional Statements
- ✓ Functions
- ✓ Sequences

Terminology

Algorithms

A step-by-step set of operations to be performed to help solve a problem

Conditional Statements

Making decisions based on conditions i.e. if some condition is met do something, else do nothing or something else

Function

A type of procedure or routine that performs a distinct

learned that you didn't know before?"

Introduce today's project: creating interactive emojis using a programming language called Processing!

Have learners go to <https://www.openprocessing.org/> and sign in using the class account that you created.

Direct learners to the 'Getting Started' sketch in OpenProcessing:

<https://www.openprocessing.org/sketch/561189>

Have them 'Create a Fork' of the project, then 'Edit' and rename their sketch (see Step 1 of solution sheet for guidance)

Point out the following elements of the 'Getting Started' sketch:

Void setup: This sets up our sketch. It is the first thing that happens when we run our project, and only happens once.

Void draw: This is where we add our code for drawing and animations. This happens next, but loops over and over, in the same order (or sequence) that the code is listed.

Semicolons: We need to add these after each line of code that we add inside of the void and draw functions.

Using x,y coordinates: Note that our canvas uses x,y coordinates... HOWEVER (0,0) is in a different location than we are used to. Instead of being in the middle, (0,0) is in the top, left corner. When X increases, it moves right. When Y increases, it moves down.

Comments: The //backslashes are comments - these are for our eyes only.

Introduce learners to their new BFF - the Processing Reference! Have them open it in another tab: <https://processing.org/reference/>

Go through 2-3 challenges:

- Move the circle to the middle of your sketch.

operation. There are often 'canned' functions that exist already

Sequences

Identifying a series of steps for a task. Computers and Scratch read and perform commands in order from top to bottom

Curricular Connections

Colour, Colour models (RGB), Colour theory, Shape and form, Symbols, Alignment, Measurement, Area, Graphing, x,y coordinates, Geometry, 2D shapes

- DRAW a rectangle BEHIND your circle (try to make a watch!)
- Change the colour of the background (hint: this only needs to happen once)

See this sketch for an example solution: <https://www.openprocessing.org/sketch/561200>

Activity

Show learners the example project (<https://www.openprocessing.org/sketch/559871>).
We're creating emojis using code!

Have learners open the Starter Project (<http://bit.ly/emoji-starter>) and click on the "FORK" icon > "Create a Fork"

Use the Solution Sheet to guide learners through the following steps:

- Mapping shapes to coordinates
- Drawing more shapes
- Adding interactivity
- Adding colour using RGB values

Give learners time to work on their sketches. If you have time, include "Add-On: Customize Your Emoji."

Assessment

Learning Outcomes

I can create algorithms in Processing
I can position shapes using x,y coordinates
I can use sequences to layer shapes in my sketch
I can use conditionals to make my project interactive
I can digitally mix colours using RGB values

Success Criteria

I renamed my copy of the starter project using my first name
All of the starting shapes are repositioned onto my emoji's face
I added at least one additional shape to my sketch
My project is interactive (the user can make something happen by clicking, pressing, or moving)
I changed the colour of at least one element in my sketch

Assessment Ideas

Have learners write an Artist's Statement to explain their design decisions, and add it into their sketch as a comment. "This artwork is titled [name]. It is a [style of art] using [materials]. I chose [design decision] because [reason]." --> E.g. "This artwork is titled Playful. It is a generative art project created using Processing. I chose to make the background blue because it is a complementary colour to yellow."

Have learners add comments to their code for each change/addition they make to the starter project to ensure that they understand what each line of code does.

Extension

Watch video Step into The Page about Glen Keane (animator at Disney) here: bit.ly/step-into-the-page. Practice drawing emotions the same way that Glen Keane and his father did (01:30) - by drawing circles on a paper with a list of expressions under each. Have learners choose one of these faces as inspiration for their emoji art.

Link to a math lesson on Area, where learners record the surface area of each circle and rectangle on their sketch, in pixels. Challenge them to add up the total surface area of all of the circles and rectangle drawn in their sketch, or calculate the total surface area of a partner's processing sketch.

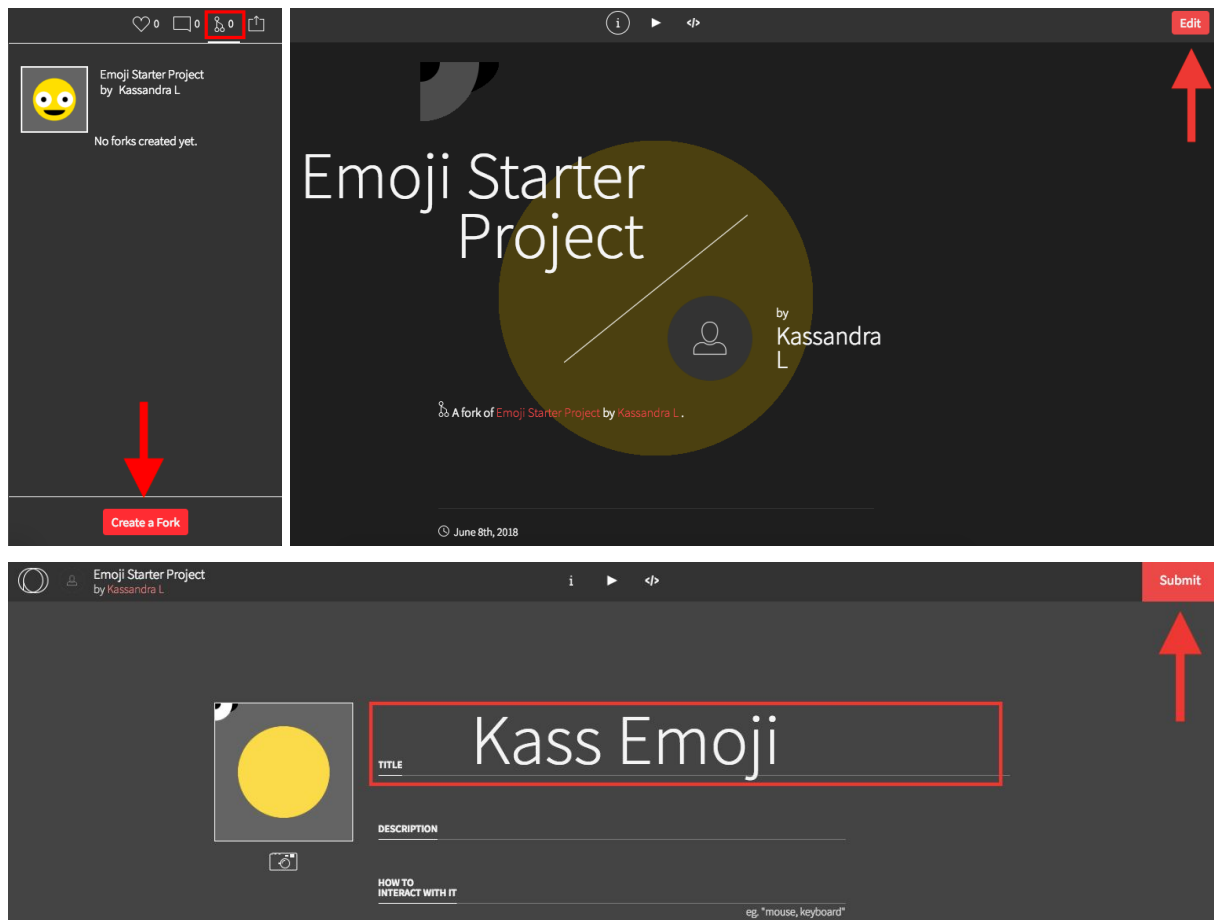
Include a focus on colour theory. Discuss RGB as an additive colour model, use Adobe Colour to learn about colour schemes, or discuss colour + emotion.

Use this opportunity to discuss when learners should and should not be using emojis at school (e.g. Emojis should never appear in essays, but are okay to include when brainstorming in Google Docs).

Code-moji

STEP 1: Open the Starter Project

1. Sign in to openprocessing.org
2. Open the starter project: <http://bit.ly/emoji-starter>
3. Fork the project & change the project name

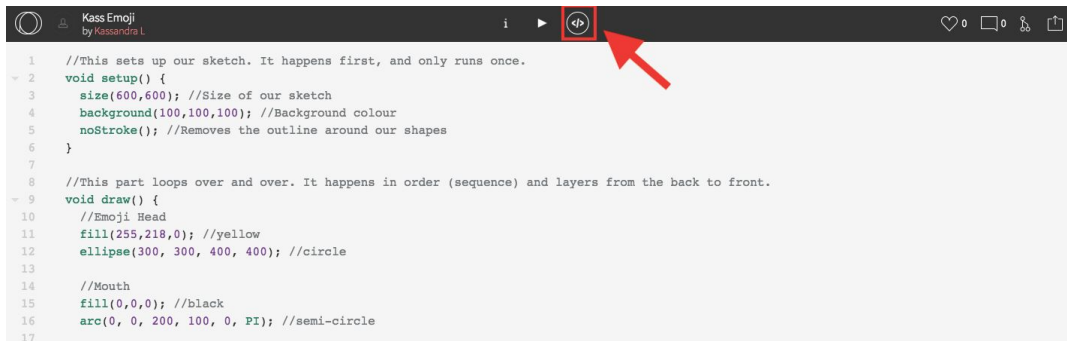


STEP 2: Mapping Shapes to Coordinates

1. Click on the Code icon `</>`
2. Take a look at the code that is already here. There's a Head, Eye (eyeball + pupil), and Mouth. We have a few parts of our emoji already existing - we just need to move them to the correct place.
3. Review X,Y Coordinates using the Stretch & Scuttle unplugged activity: <https://www.canadalearningcode.ca/lessons/stretch-and-scuttle/>

4. Move the Mouth: change the x,y values of the arc
5. Move the Eye: change the x,y values of the ellipse
6. Move the Pupil (on top of the white eyeball): change the x,y values of the ellipse
7. Don't forget to save! (top, right corner)

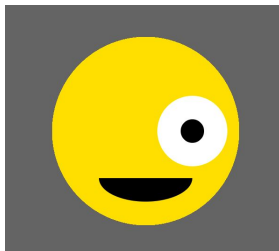
Note: After changing the code, model the importance of testing by continuously clicking back and forth between the Code icon and Run icon.



```

1 //This sets up our sketch. It happens first, and only runs once.
2 void setup() {
3   size(600,600); //Size of our sketch
4   background(100,100,100); //Background colour
5   noStroke(); //Removes the outline around our shapes
6 }
7
8 //This part loops over and over. It happens in order (sequence) and layers from the back to front.
9 void draw() {
10  //Emoji Head
11  fill(255,218,0); //yellow
12  ellipse(300, 300, 400, 400); //circle
13
14  //Mouth
15  fill(0,0,0); //black
16  arc(0, 0, 200, 100, 0, PI); //semi-circle
17

```



```

14  //Mouth
15  fill(0,0,0); //black
16  arc(300, 400, 200, 100, 0, PI); //semi-circle
17
18  //Right Eye
19  fill(255,255,255); //white
20  ellipse(400, 300, 150, 150); //circle
21
22  //Right Pupil
23  fill(0,0,0); //black
24  ellipse(400, 300, 50, 50); //circle
25

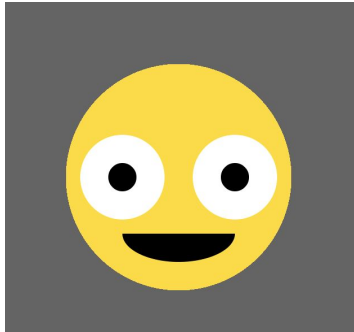
```

STEP 3: Drawing More Shapes

We need to complete our emoji! We can either (a) Copy the eye code and move it over, or (B) Draw a new shape using the Reference

(A) To copy:

1. Select the eye + pupil code and COPY (ctrl+c) then PASTE (ctrl+v) the code inside of 'draw'
2. Change the x values so there are two eyes, side-by-side



```

26 //Left Eye
27 fill(255,255,255); //white
28 ellipse(200, 300, 150, 150); //circle
29
30 //Left Pupil
31 fill(0,0,0); //black
32 ellipse(200, 300, 50, 50); //circle
33

```

(B) To draw a new shape:

1. Find a new shape using the reference - let's try using the arc to make a winking face
2. Type or paste the arc code inside of 'draw'
3. Fill in the values of the shape. For the winking shape, we need to start at PI and end at TWO_PI (instead of starting at 0 and ending at PI like our mouth)
4. Don't forget to save!



```

29 //Left eye - winking
30 //also black - don't need to update fill()
31 arc(200, 300, 100, 50, PI, TWO_PI); //semi-circle
32

```

STEP 4: Adding Interactivity

Making our project **interactive** means giving the user the ability to change and interact with our sketch. As an example - let's make something happen IF the user presses a key on the keyboard

Note: IF is a conditional - we are checking IF something is true. On the condition that is true, we will tell the program to do something.

1. Open the **keyPressed** page of the Reference to see how it works (the one without the parentheses) <https://processing.org/reference/keyPressed.html>
2. We just need the IF statement (see below) - type or copy+paste the code inside of the draw function
3. Add something inside of the conditional statement - this will only be drawn IF a key is pressed. For example, we'll draw a tongue.
4. Don't forget to save!


```

Name      keyPressed

Examples  // Click on the image to give it focus,
          // and then press any key.

          // Note: The rectangle in this example may
          // flicker as the operating system may
          // register a long key press as a repetition
          // of key presses.

void draw() {
  if (keyPressed == true) {
    fill(0);
  } else {
    fill(255);
  }
  rect(25, 25, 50, 50);
}
    
```

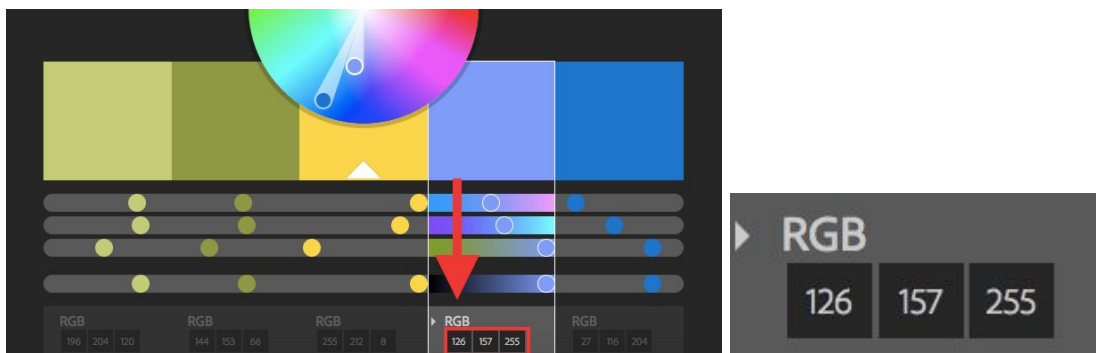


```

30      //Sticking out tongue - IF a key is pressed
31      if(keyPressed == true){
32          fill(255,78,83); //red
33          arc(300, 400, 100, 150, 0, PI); //semi-circle
34      }
    
```

STEP 5: Adding Colour using RGB values

1. Open the background page in the Reference to see how it works
Note: the three values in the parentheses are RGB (Red, Green, Blue)
2. Have learners go to color.adobe.com
3. Select a colour for the background
4. Replace the existing RGB values for the background. E.g. Blue:
5. Do the same for any fill() values to change the colour of the emoji itself
6. Don't forget to save!




```
4 void setup() {  
5   size(600,600); //Size of our sketch  
6   background(126,157,255); //Background colour  
7   noStroke(); //Removes the outline around our shapes  
8 }
```

ADD-ON: Customize Your Emoji

1. Give learners time to further customise their emoji. Challenge them to either code one of the following expressions, or create their own, new expression.
2. Note: Some emojis are more difficult than others, but the question is never "CAN I make this" - It's "HOW can I make this?" - We can create anything, but need to use the reference, work with classmates, and use our Googling skills to find a way.



Worried



Straight face



Rolling eyes



Happy (blushing)