Processing Cheat Sheet – Loops, Modulus

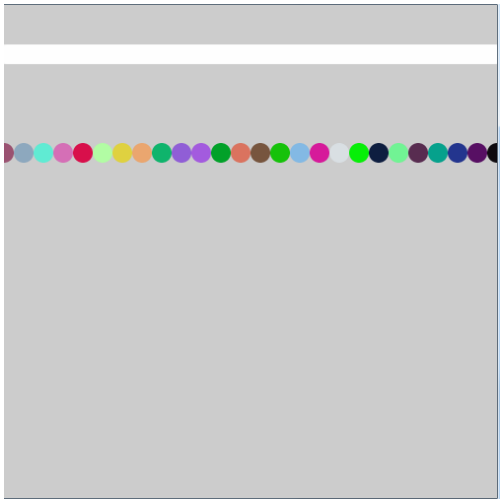
1 – for Loops and Modulus

Loops let you repeat a certain piece of code, to save you time and energy, and accomplish complex things that would otherwise be impossible.

The modulus (%) operator gives the remainder when dividing two numbers. This means we can look for a modulus (%) of zero to check if a number is divisible by another.

The loop below repeats ellipses across the screen, drawing one row without any spacing, and another row with an ellipse every 20 pixels:

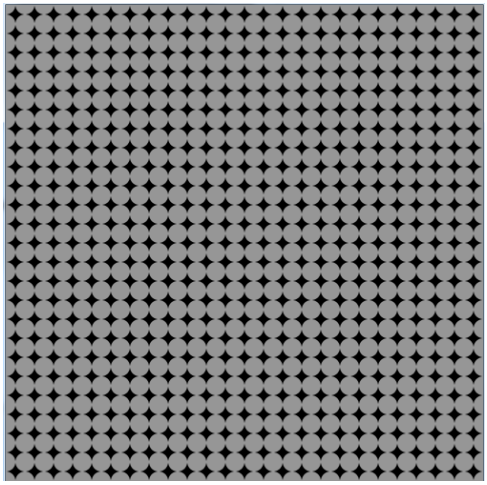
```
function setup(){
createCanvas(400, 400);
var counter;
//draws ellipses across screen
for (counter=0; counter <=width; counter++){
      noStroke();
      fill(255);

      ellipse(counter, 50, 20, 20);

      fill(random(255), random(255), random(255));

      //draw an ellipse if 'counter' is divisible by 20
      if(counter%20 == 0) ellipse(counter, 150, 20, 20);
     }
}
```



2 – Nested for Loops

By putting a loop inside a loop, we can traverse the screen both horizontally and vertically.

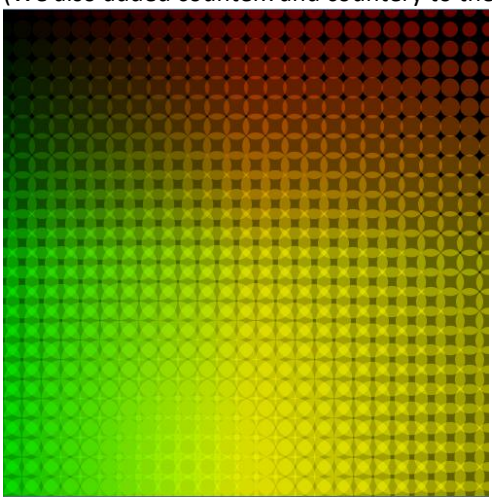
```
//draws ellipses across AND DOWN screen
//noprotect

function setup(){
createCanvas(400, 400);
background(0);

var counterx;
var countery;

//loop that moves down screen vertically
//to draw each line of dots
for (countery=0; countery <=height; countery++){

      //loop that moves across each line, drawing dots
      for (counterx=0; counterx<=width; counterx++){
            if(counterx%20 == 0 && countery%20==0){
                  fill(255, 150);
                  noStroke();
                  ellipse(counterx, countery, 20, 20);
                 }
            }
      }
}
```



3 – Nested for Loops + Mouse Interaction

We can use mouseX and mouseY here, like we would any other code, to affect things like shape size, position, and color. Here, we'll use a function called **dist()** to measure the distance between the mouse and each ellipse, and change the ellipse size based on mouse closeness.

(We also added counterx and countery to the color values to make things more visually interesting!)



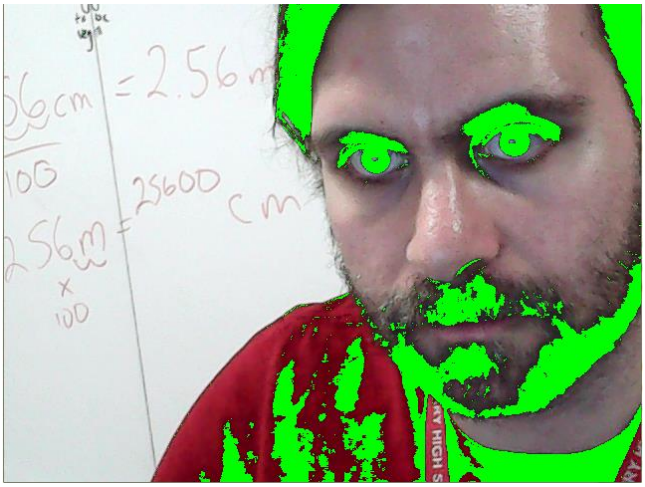
```
//noprotect

function setup(){
createCanvas(400, 400);
}

function draw(){
background(0);

var counterx;
var countery;

for (countery=0; countery<=height; countery++){
      for (counterx=0; counterx<=width; counterx++){

            if (counterx%20 == 0 && countery%20==0){

            fill(counterx, countery, 0, 100);
            noStroke();

            //create a variable called 'distance' that
            //contains the distance between the mouse
            //and the current ellipse being drawn...

            var distance = dist (counterx, countery, mouseX, mouseY);

            //...and use that value to change the value of
            //a variable called 'size' which we'll use as
            //the width and height of the ellipse

            var size = map(distance, 0, 500, 50, 5);
            ellipse(counterx, countery, size, size);
            }
      }
}
}
```

Processing Cheat Sheet –Video Input

1 – Basic Video Input

This code just create a capture feed from the camera, then captures frames from the camera, and draws them to the screen. Then the **filter()** function is used to add a posterize effect.

```
var capture;

function setup() {
  createCanvas(640, 480);
  capture = createCapture(VIDEO);
  capture.size(640, 480);
  capture.hide();
}

function draw() {
  image(capture, 0, 0, 640, 480);
  filter(POSTERIZE,3); // add a visual effect
}
```

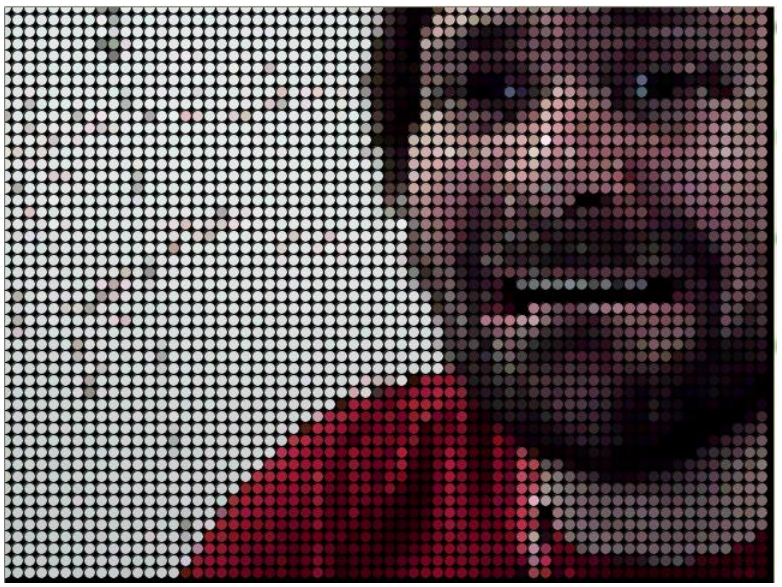


2 – Extracting and Altering Pixel Colors

```
//noprotect
var capture;

//creates a variable to store the current video frame
var vidframe;

function setup() {
  createCanvas(640, 480);
  capture = createCapture(VIDEO);
  capture.size(640, 480);
  capture.hide();

  //set the size of the 'vidframe' to store frames captured by 'capture'
  vidframe = createGraphics(640, 480);
}

function draw() {
  //draws the captured frame into our 'vidframe' image
  vidframe.image(capture, 0, 0, 640, 480);

  //creates an array of pixel color information that we can tweak and change
  vidframe.loadPixels();
  var counter;// this variable will take us through the loop to edit each pixel

  //use a for loop to travel through the array of pixel data and make changes
  for (counter = 0; counter < vidframe.width*vidframe.height; counter++){

    //grab the colors for the current pixel
    var r = vidframe.pixels[counter*4]; //store the red value of this pixel
    var g = vidframe.pixels[counter*4+1]; //store the green value of this pixel
    var b = vidframe.pixels[counter*4+2]; //store the blue value of this pixel
    //check the total color values of the current pixel
    //If it's dark, replace the pixel with neon green!
    if (r+g+b < 150){
       vidframe.pixels[counter*4] = 0; //the blue value
       vidframe.pixels[counter*4+1] = 255; //the green value
     vidframe.pixels[counter*4+2] = 0; //the red value
    }
 }

  //updatePixels() locks in any changes we made to the pixels
  vidframe.updatePixels();

  //draw our "vidframe" image onto our canvas
  image (vidframe, 0, 0, 640, 480);
}
```

3 – Extracting Pixel Colors and Using them in Shapes

This example extracts the color values of the pixels, like above, but uses them to color shapes, which are drawn using a loop and modulus, like our earlier loop projects.



```
//noprotect

var capture;
var vidframe;

function setup(){
  createCanvas(640, 480);
  capture = createCapture(VIDEO);
  capture.size(640, 480);
  capture.hide();
  vidframe = createGraphics(640, 480);
}

function draw(){

  background(0);
  //draws the captured frame into our 'vidframe' image
  vidframe.image(capture,0,0,640,480);

  //creates an array of pixel color information that we can tweak and change
  vidframe.loadPixels();

  var counter;

  //Use a loop to travel through every pixel of the video image.
  //Notice we're counting up 10 at a time in these loops
  //That's because we don't need to look at every pixel,
  //just every 10th pixel, so we can draw a circle.
  for (counter = 0; counter < width*height; counter +=10){

      //Use the modulus in an 'if' statement to make sure
      //we draw ellipses only every 10 rows
      if (int(counter/width)%10==0){ //IF we're on row 0, 10, 20, 30, etc...
         var r = vidframe.pixels[counter*4]; //store red value of this pixel
         var g = vidframe.pixels[counter*4+1];//store green value of this pixel
         var b = vidframe.pixels[counter*4+2];//store blue value of this pixel

         // use that color to draw next shape
         fill(color(r,g,b));

         // calculate what 'column' we're on
         // by taking the modulus of our current
         // position in the pixel array over
         // the width of the screen
         var column = int(counter % width);

         // calculate what 'row' we're on
         // by dividing our current
         // position in the pixel array by
         // the width of the screen
         var row = int(counter / width);

         //draw the 10x10 circle at the pixel position
         ellipse(column, row, 10, 10);
      }
  }
}
```