Universidad De Guadalajara

Centro Universitario De Ciencias Exactas E Ingenierías

División De Electrónica Y Computación

Departamento De Ciencias Computacionales

HIPERMEDIA NRC: 154781 SEC: D07 2019B

Act02 - De la creatividad en una imagen digital

Profesor: LOPEZ CISNEROS, JUAN JOSE

Nombre: Ruiz Ortiz Valentín Alejandro. Codigo:213330441

Descripción de la actividad

Se llevará a desarrollo una aplicación en donde se muestra los principios del uso de Canvas y Processing para el desarrollo de una imagen en a cuál será diseñada por totalmente el medio digital a través de las diferentes tecnologías implementadas ya mencionadas.

Objetivo (s) de la actividad

El objetivo de esta actividad es comparar las diferentes tecnologías mencionadas, así como sus similitudes pues al ser dos herramientas del mundo tecnológico para el desarrollo de imágenes o mejor llamado multimedia son herramientas que pueden llegar hacer lo mismo pero con diferencias abismales en cuanto a su desarrollo.

Fundamentos Teóricos Estudiados

Para poder empezar a comprender las tecnologías primero debemos comprender que son las tecnologías como lo mencionaremos en el apartado mas adelante.

¿Qué es processing?

Al mencionar a processing es un software de código abierto que cualquier persona puede usar donde las interfaces son creadas gracias a java. Su descarga es relativamente sin alguna complejidad. Su estructura de los elementos dentro del mismo es muy fácil.

Al parecido que otros lenguajes de programación tiene la facilidad de imprimir elementos en pantalla, pero esta vez nos referimos a la consola donde la forma de imprimir es con println (“”)

Al igual que la forma de poner algún tipo de comentarios basta con doble líneas o línea y asterisco.

La forma de crear las funciones es algo similar solo debemos expresar el tipo de dato a devolver, en este caso existen todos los tipos de datos al igual que en lenguajes como java.

Cabe resaltar que es un lenguaje donde la sensibilidad al igual que otros es muy grande.

En processesing existen dos funciones que son de os mas importantes en el programa donde tenemos a setup () y a la función de draw () en la primera usamos todo lo necesario para inicializar en la siguiente función se llamara de forma iterativa una tras otra vez en la cual se entiende por ende que se realizara el dibujado.

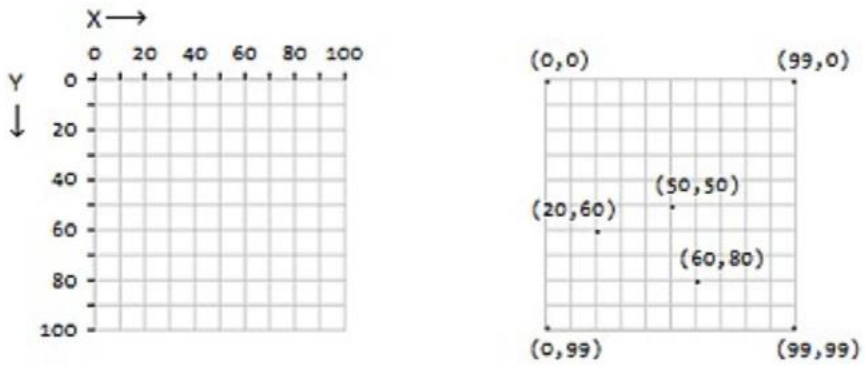


Imagen ilustrativa de distribucion de los pixeles.

-Figuras primitivas

Un punto es el elemento visual más simple y se dibuja con la función point(): point(x,y)

Esta función tiene dos parámetros: el primero es la coordenada x y el segundo es la coordenada y. A menos que se especifique otra cosa, un punto es del tamaño de un sólo píxel.

```
point(20, 20);
point(30, 30);
point(40, 40);
point(50, 50);
point(60, 60);
```



```
point(-500, 100); //Los parámetros negativos no provocan
point(400, -600); //error, pero no se verán en la ventana de
point(140, 2500); //representación.
point(2500, 100);
```

Imagen ilustrativa de creación de puntos

Es posible dibujar cualquier línea mediante una serie de puntos, pero son más simples de dibujar con la función line(). Esta función tiene cuatro parámetros, dos por cada extremo:

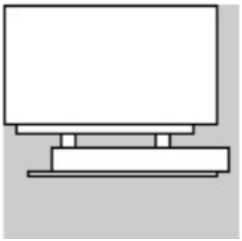
```
line(x1, y1, x2, y2)
```

Imagen Ilustrativa de creación de línea

Los primeros dos parámetros establecen la posición donde la línea empieza y los dos últimos establecen la posición donde la línea termina.

```
line(25, 90, 80, 60);
line(50, 12, 42, 90);
line(45, 30, 18, 36);
```

Creación de línea con ejemplo practico

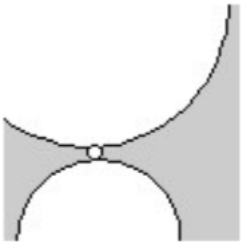


```
rect(0, 0, 90, 50);
rect(5, 50, 75, 4);
rect(24, 54, 6, 6);
rect(64, 54, 6, 6);
rect(20, 60, 75, 10);
rect(10, 70, 80, 2);
```

Creación de cuadro

La función ellipse() dibuja una elipse en la ventana de representación: ellipse(x, y, ancho, alto)

Los dos primeros parámetros establecen la localización del centro de la elipse, el tercero establece la ancho, y el cuarto la altura. Use el mismo valor de ancho y de alto para dibujar un círculo.



```
ellipse(35, 0, 120, 120);
ellipse(38, 62, 6, 6);
ellipse(40, 100, 70, 70);
```

Creación de elipse

¿Que es Canvas?

Etiqueta o elemento en HTML5 que permite la generación de gráficos en forma dinámica por medio de programación dentro de una página.

Posee dos atributos width (ancho) y height (alto), el tamaño por defecto es 150.

Permite generar gráficos 2D, juegos, animaciones y composición de imágenes

SVG es otra etiqueta que cumple con funciones similares

El elemento <canvas>

Este elemento genera un espacio rectangular vacío en la página web (lienzo) en el cual serán mostrados los resultados de ejecutar los métodos provistos por la API. Cuando es creado, produce sólo un espacio en blanco, como un elemento vacío, pero con un propósito totalmente diferente.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Canvas API</title>
  <script src="canvas.js"></script>
</head>
<body>
  <section id="cajalienzo">
    <canvas id="lienzo" width="500" height="300">
      Su navegador no soporta el elemento canvas
    </canvas>
  </section>
```

Los atributos width (ancho) y height (alto) declaran el tamaño del lienzo en pixeles. Estos atributos son necesarios debido a que todo lo que sea dibujado sobre el elemento tendrá esos valores como referencia. Al atributo id, como en otros casos, nos facilita el acceso al elemento desde el código Javascript.

Eso es básicamente todo lo que el elemento <canvas> hace. Simplemente crea una caja vacía en la pantalla. Es solo a través de Javascript y los nuevos métodos y propiedades introducidos por la API que esta superficie se transforma en algo práctico.

getContext()

El método getContext() es el primer método que tenemos que llamar para dejar al elemento <canvas> listo para trabajar. Genera un contexto de dibujo que será asignado al lienzo. A través de la referencia que retorna podremos aplicar el resto de la API.

```
function iniciar(){
  var elemento=document.getElementById('lienzo');
  lienzo=elemento.getContext('2d');
}
window.addEventListener("load", iniciar, false);
```

Dibujando rectángulos

fillRect(x, y, ancho, alto) Este método dibuja un rectángulo sólido. La esquina superior izquierda será ubicada en la posición especificada por los atributos x e y. Los atributos ancho y alto declaran el tamaño.

strokeRect(x, y, ancho, alto) Similar al método anterior, éste dibujará un rectángulo vacío (solo su contorno).

clearRect(x, y, ancho, alto) Esta método es usado para substraer pixeles del área especificada por sus atributos. Es un borrador rectangular.

```
function iniciar(){
  var elemento=document.getElementById('lienzo');
  lienzo=elemento.getContext('2d');
  lienzo.strokeRect(100,100,120,120);
  lienzo.fillRect(110,110,100,100);
  lienzo.clearRect(120,120,80,80);
}
window.addEventListener("load", iniciar, false);
```

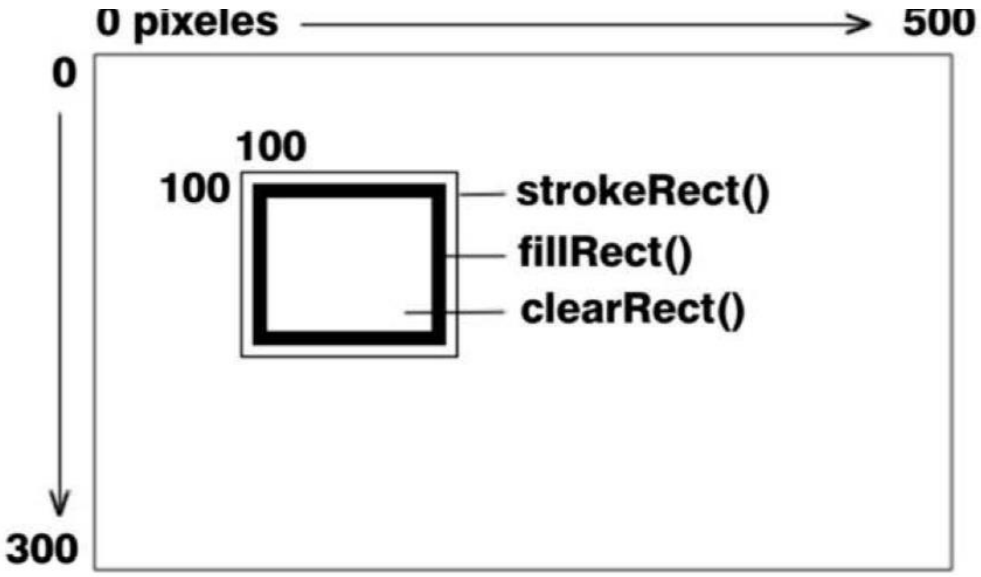


Imagen ilustrativa de pixeles en Canvas.

Creando trazados

beginPath() Este método comienza la descripción de una nueva figura. Es llamado en primer lugar, antes de comenzar a crear el trazado.

closePath() Este método cierra el trazado generando una línea recta desde el último punto hasta el punto de origen. Puede ser ignorado cuando utilizamos el método fill() para dibujar el trazado en el lienzo.

También contamos con tres métodos para dibujar el trazado en el lienzo:

stroke() Este método dibuja el trazado como una figura vacía (solo el contorno).
fill() Este método dibuja el trazado como una figura sólida. Cuando usamos este método no necesitamos cerrar el trazado con closePath(), el trazado es automáticamente

cerrado con una línea recta trazada desde el punto final hasta el origen.
clip() Este método declara una nueva área de corte para el contexto. Cuando el contexto es inicializado, el área de corte es el área completa ocupada por el lienzo. El método clip() cambiará el área de corte a una nueva forma creando de este modo una

máscara. Todo lo que caiga fuera de esa máscara no será dibujado.

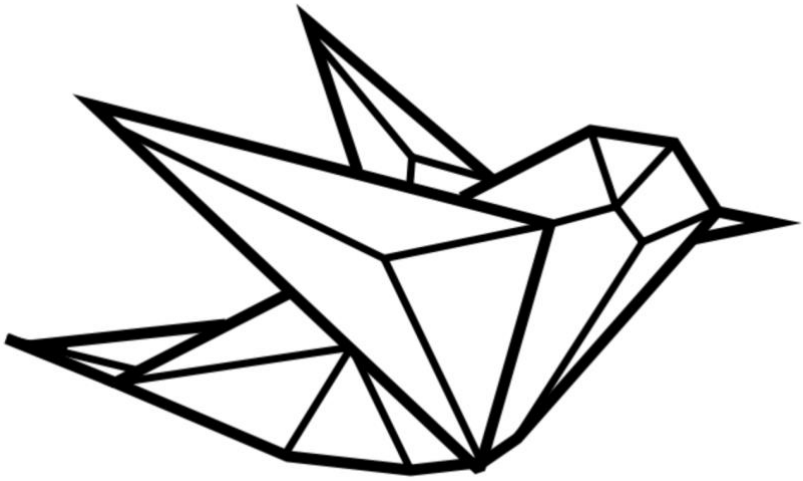
moveTo(x, y) Este método mueve el lápiz a una posición específica para continuar con el trazado. Nos permite comenzar o continuar el trazado desde diferentes puntos, evitando líneas continuas.

lineTo(x, y) Este método genera una línea recta desde la posición actual del lápiz hasta la nueva declarada por los atributos x e y.

rect(x, y, ancho, alto) Este método genera un rectángulo. A diferencia de los métodos estudiados anteriormente, éste generará un rectángulo que formará parte del trazado (no directamente dibujado en el lienzo). Los atributos tienen la misma función.

arc(x, y, radio, ángulo inicio, ángulo final, dirección) Este método genera un arco o un círculo en la posición x e y, con un radio y desde un ángulo declarado por sus atributos. El último valor es un valor booleano (falso o verdadero) para indicar la dirección a favor o en contra de las agujas del reloj.

Practica



Se realizo el siguiente dibujo a través de Canvas.

```
<!-- Learn about this code on MDN:
https://developer.mozilla.org/es/docs/Web/G
de/HTML/Canvas_tutorial/Basic_usage -->

<html>
  <head>
    <title>Canvas tutorial</title>
    <script type="text/javascript">
    </script>
    <style type="text/css">
      canvas { border: 1px solid black; }
    </style>
  </head>
  <body onload="draw();">
    <canvas id="tutorial" width="844"
height="932"></canvas>
  </body>
</html>
```

Este es el código que se implemento para llevar a cabo el dibujo presentado anteriormente.

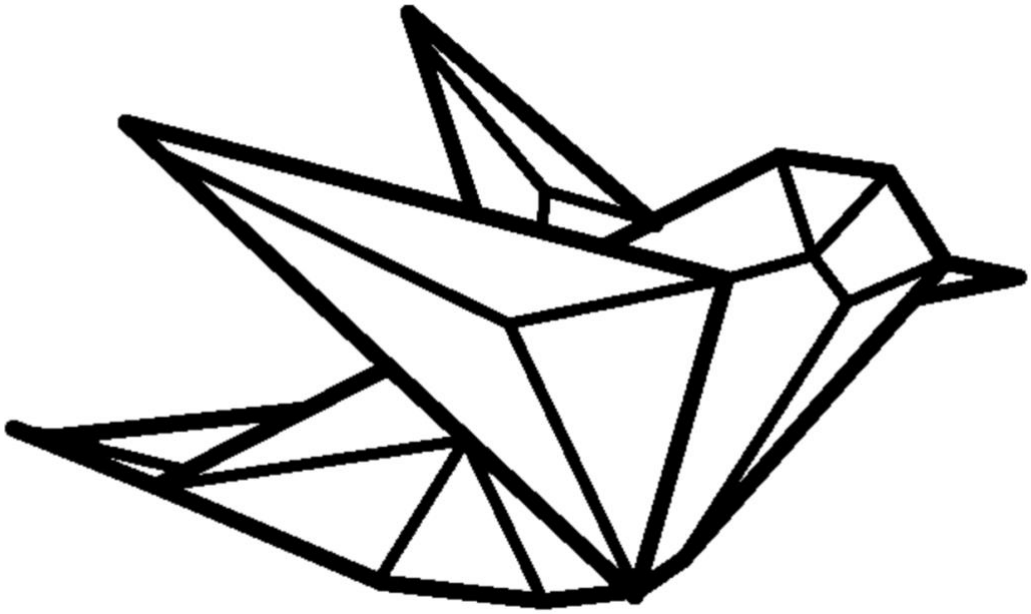
```
function draw(){
        var canvas = document.getElementById('tutorial');
        if (canvas.getContext){
          var ctx = canvas.getContext('2d');

          ctx.lineWidth = 9;

          //ala
          ctx.beginPath();
          ctx.moveTo(200,348);
          ctx.lineTo(561,442);
          ctx.lineTo(506,633);
          ctx.lineTo(503,634);
          ctx.closePath();
          ctx.stroke();

          //cuerpo
          ctx.beginPath();
          ctx.moveTo(506,631);
```

Este es un fragmento del código para la realización del dibujo.



Este es el dibujo con el otro tipo de desarrollo como lo es Processing.

```
function setup() {
  createCanvas(844, 932);
}

function draw() {

  strokeWeight(9);

  //ala
  line(200,348, 561,442);
  line(561,442, 506,633);
  line(506,633, 200,348);

  //cuerpo
  line(506,631, 535,611);
  line(535,611, 693,432);
  line(693,432, 659,378);
  line(659,378, 593,368);
  line(593,368, 491,421);
```

Este es un fragmento del código para la realización de dibujo.

Conclusión

Esta practica podemos llegar a ver lo siguiente que es mas diferente y similar Canvas y processing una vez comprendido cada uno concluimos que para practicas me gusta mas en lo personal el programa processing pues al ser mas matemático podemos realizar mucho mas tipo de cuestiones a diferencia de Canvas, en esta practica no se tuvo ningún tipo de problema pues en verdad fue muy sencillo.