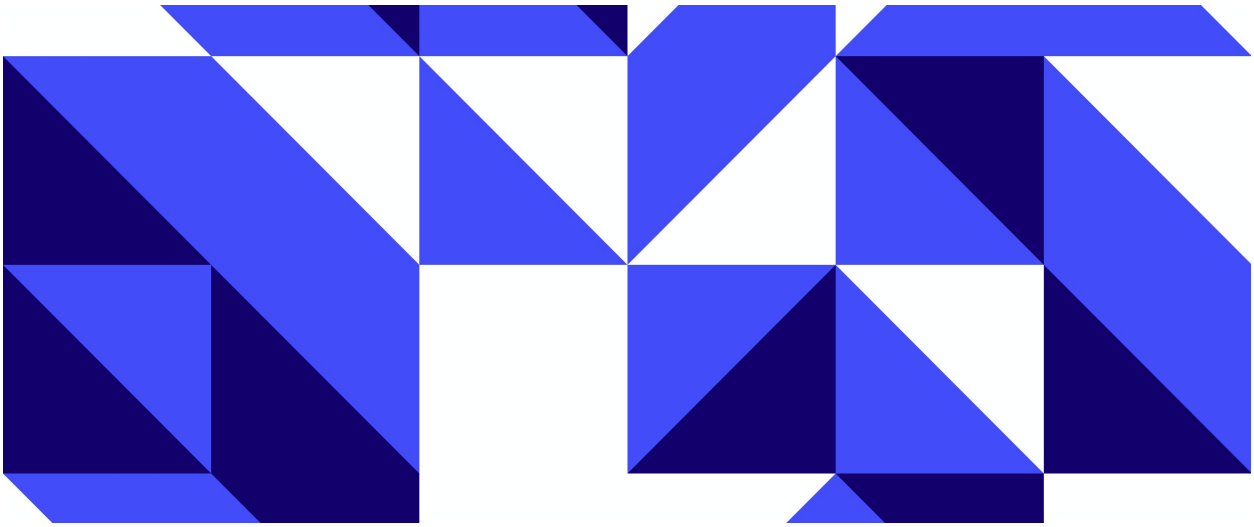


Proyecto 02: Identificar figuras geométricas en imágenes

1.- Definición del problema

Entendimiento del problema

Arsenal

Paradigma: Orientado a Objetos.

Lenguaje de programación utilizado: C#

Requisitos funcionales (qué debe hacer el programa)

Entrada: Imagen de tipo BMP.

Salida: Mensaje en terminal donde indica el tipo de figura junto con su color en hexadecimal.

Requisitos no funcionales (cómo debe comportarse el programa)

Eficiencia

Se espera que el programa tenga una complejidad cuadrática sobre el número de pixeles de la imagen, pues se trata a la imagen como un arreglo de dos dimensiones.

Tolerancia a fallas.

Debido a que es un programa que se corre en terminal, cuyo único parámetro es la ruta del archivo BMP, existen muy pocos errores posibles.

Amigabilidad

Es necesario que el usuario tenga un conocimiento mínimo sobre el uso de la terminal. Así también debe tener conocimiento sobre código RGB para interpretar la salida del programa.

Escalabilidad

Podemos escalar el programa agregando una interfaz gráfica de usuario que le permita visualizar la imagen y el resultado arrojado por el programa, en lugar de mostrar el programa en terminal.

También podemos mejorar el programa mostrando el nombre del color y no únicamente su código RGB.

Seguridad

Al no estar manipulando datos sensibles, es muy poco necesario tener un apartado de seguridad.

Interoperabilidad

Siguiendo el principio de encapsulación, nuestro programa está separado por clases, que se delimitan a hacer una labor específica.

2.- Análisis del problema

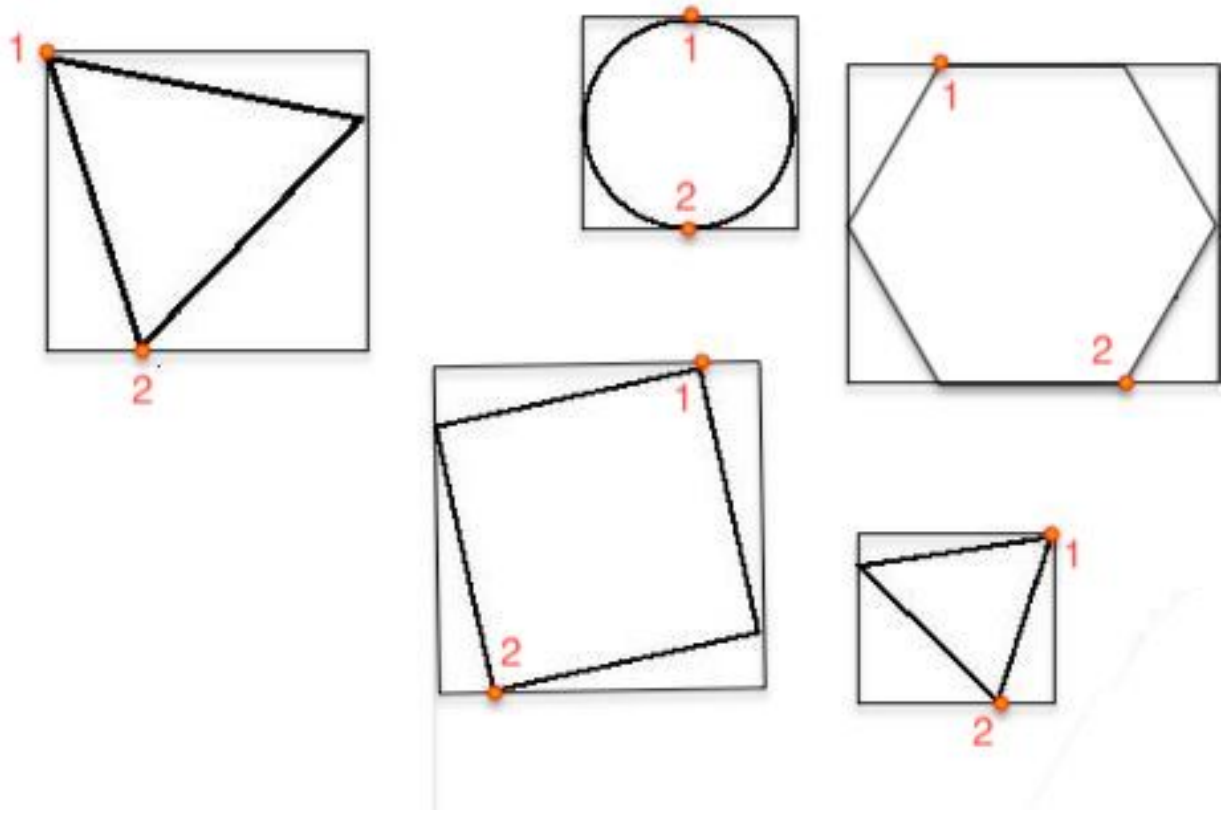
Se requiere realizar un programa para devolver los datos de cada figura que haya en una imagen. Así, al obtener la imagen como entrada, queremos primero distinguir las localizaciones de las figuras en la imagen recurriéndola como un arreglo. Una vez realizado, debemos delimitar la imagen, de tal forma que “recortemos” cada figura para tratar con ella sin que el resto de la imagen cause ruido. Después, debemos encontrar el centro de la figura. Posteriormente obtenemos el borde de la figura, y con ello

calculamos distancias del centro a los puntos en el borde para que nos ayude a encontrar la cantidad de máximos que hay en un conjunto de distancias. Las distancias mayores serán los vértices, y dependiendo de la cantidad de máximos, se tratará de una figura u otra. Por ejemplo, si la figura tiene 4 máximos, se tratará de un cuadrilátero. Para obtener un resultado más preciso, optamos por promediar entre los elementos del conjunto de distancias para tener máximos más definidos y facilitar la búsqueda de los mismos.

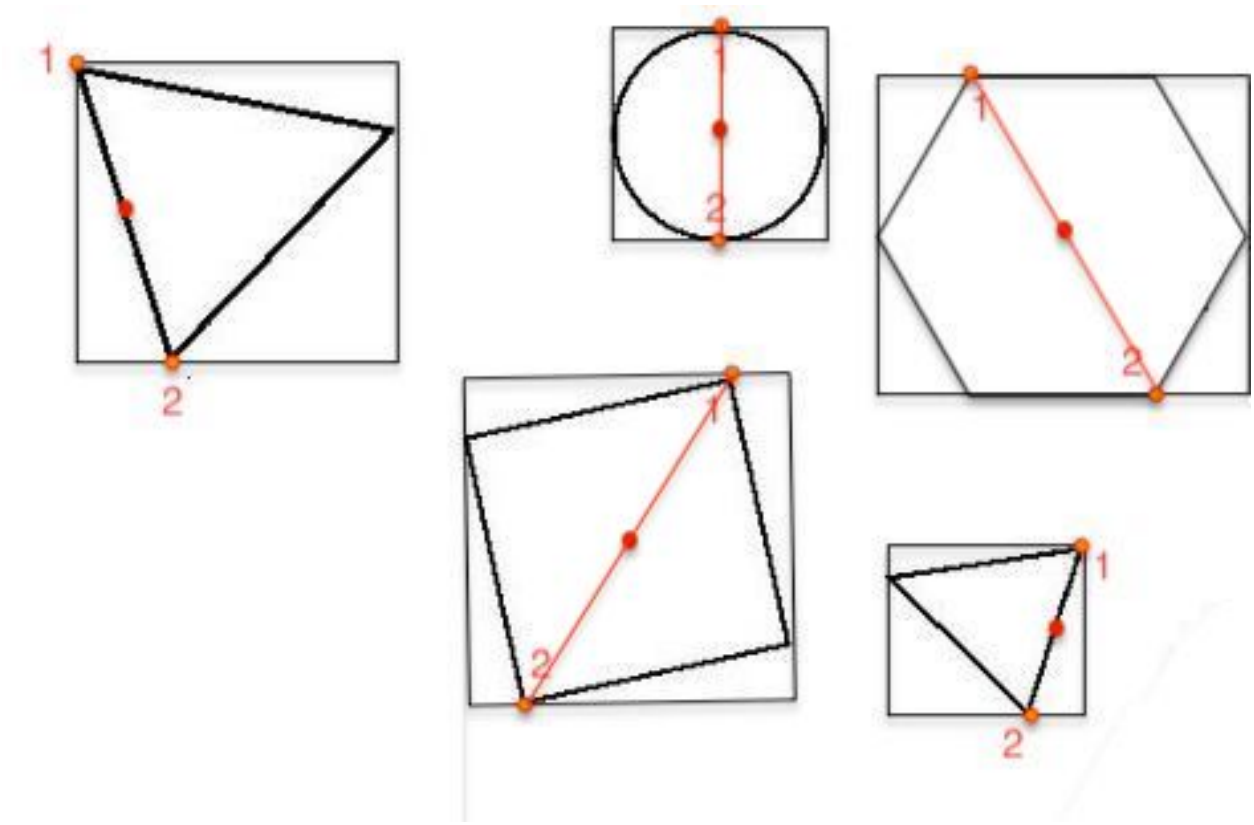
Observaciones:

Dado a que la imagen se manipula como un arreglo bidimensional, se preservará un orden, siendo que el primer y último pixel de la figura “recortada” son vértices.

Ejemplo:



Después obtenemos los puntos en donde se encuentra el borde de la figura (límites) con su respectivo centro que en este caso lo realizamos de la siguiente forma.



Con base en esto, no es difícil ver que en el caso del triangulo, el centro se encuentra en el borde, por lo que bastará comprobar si el punto está solo en el borde para saber que es un triángulo.

3.- Selección de la mejor alternativa

Dado que vamos a trabajar con imágenes, decidimos usar el lenguaje de programación C# puesto que este tiene integrado dentro de sí las herramientas necesarias para procesarlas, pues en otros lenguajes de programación habría que instalar bibliotecas externas.

4.- Diagrama de flujo o pseudocódigo

```
###Algoritmo para encontrar el centro de la figura ya recortada.
###Vemos a la figura como un arreglo de pixeles, se dan por hecho sus
###correspondientes getters.
Proceso EncuentraCentro(Pixel[] pixeles)
Inicio Proceso
```

```
  primerPixel <- pixeles[0]
  ultimoPixel <- pixeles[figura.lengh-1]
  medioX <- int(primerPixel.getX + ultimoPixel.getX)/2)
  medioY <- int(primerPixel.getY + ultimoPixel.getY)/2)
  Devuelve Pixel(medioX, medioY)
Fin Proceso

###Algoritmo para encontrar la distancia entre dos pixeles.
###Se dan por hecho sus correspondientes getters.
Proceso Distancia(Pixel pixel1, pixel2)
Inicio Proceso
  Devuelve sqrt(((pixel2.getX)^2 - (pixel1.getX)^2)+((pixel2.getY)^2 - (pixel1.getY)^2))
Fin Proceso

###Algoritmo que evalúa si el pixel es un borde de una figura.
Proceso EsBorde(Pixel pixel)
Inicio Proceso
  Devuelve imagen.getPixel(x,y) != pixel.getColor)
Fin Proceso
```

5.- Pensamientos a futuro

Podemos mejorar la herramienta dotándole de una interfaz gráfica amigable con el usuario, en donde se le muestre una ventana de selección de archivo, y una parte donde se le muestre la información de la imagen en un Label.

Posibles mantenimientos

Es posible mejorarlo para que tome en cuenta más figuras, tales como pentágonos, hexágonos, etc., de tal forma que al momento de leerlos, no los mande directamente a “otros”.

Cantidad a cobrar por el reto

Considerando la labor de analizar la imagen y los posibles usos del programa, vemos justa la cantidad de $30,000 pesos.