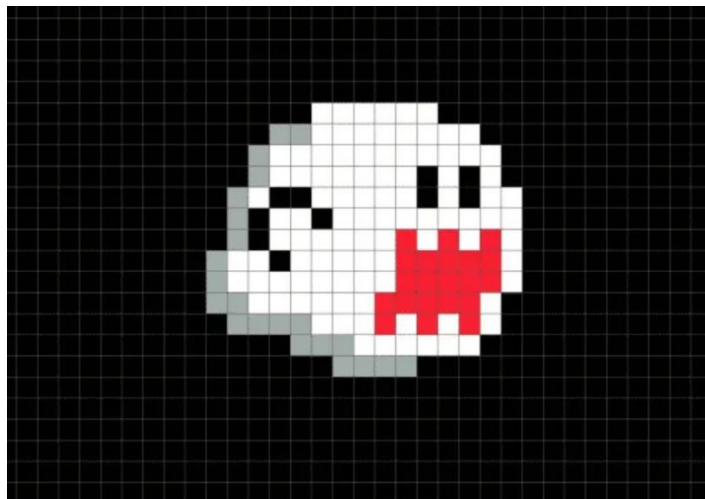
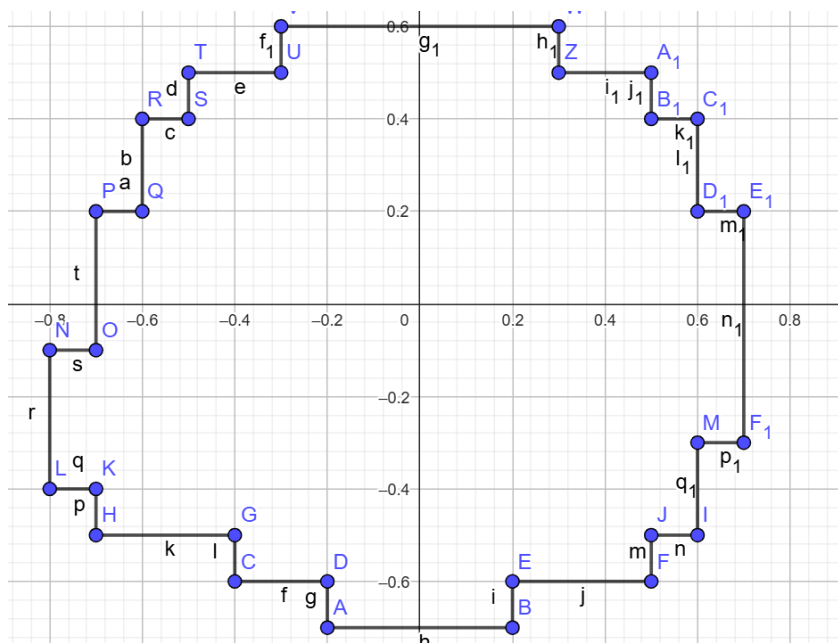


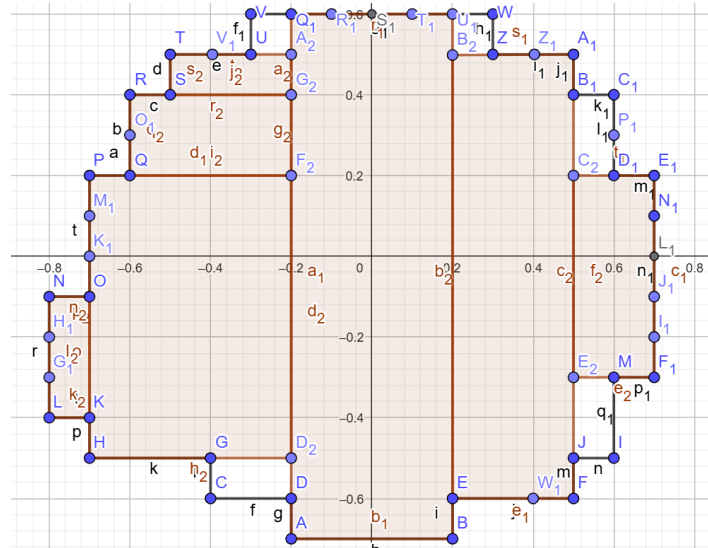
El dibujo con estilo PixelArt que se decidió replicar es el siguiente



Para entender mejor los vertices de esta imagen, se dibujo su contorno.



Posteriormente, mediante polígonos (cuadrados y rectángulos) se fue rellenando la figura. De esta manera se tienen los vértices en un contexto de -1 a 1 del cuerpo del Boo, haciendo más fácil su construcción en OpenGL.



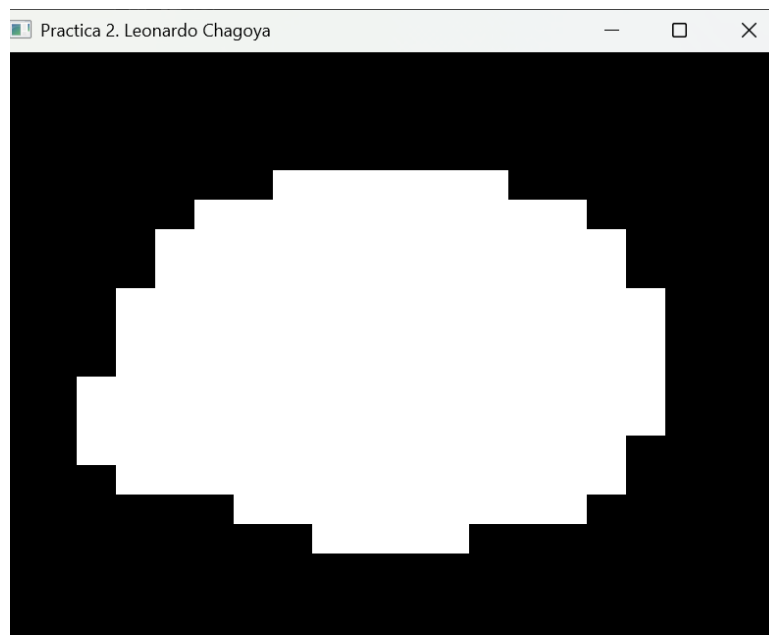
Posteriormente en el código se implementaron triángulos para la construcción de los polígonos. Se consultó la documentación de OpenGL en donde se encontró la primitiva GL_TRIANGLE_STRIP donde explica que cada grupo de 3 vértices adyacentes forma un triángulo y se invierten los índices construyendo dos triángulos a partir de 4 vértices. De esta forma solo tenemos que ordenar las coordenadas de los polígonos en GeoGebra de la siguiente manera

```
float vertices[] = {  
    //Cuerpo del Boo  
    // Rectángulo 1  
    -0.8f, -0.4f, 0.0f, 1.0f, 1.0f, 1.0f, // Abajo izquierda  
    -0.8f, -0.1f, 0.0f, 1.0f, 1.0f, 1.0f, // Arriba izquierda  
    -0.7f, -0.4f, 0.0f, 1.0f, 1.0f, 1.0f, // Abajo derecha  
    -0.7f, -0.1f, 0.0f, 1.0f, 1.0f, 1.0f, // Arriba derecha  
  
    // Rectángulo 2  
    -0.7f, -0.5f, 0.0f, 1.0f, 1.0f, 1.0f,  
    -0.7f, 0.2f, 0.0f, 1.0f, 1.0f, 1.0f,  
    -0.2f, -0.5f, 0.0f, 1.0f, 1.0f, 1.0f,  
    -0.2f, 0.2f, 0.0f, 1.0f, 1.0f, 1.0f,  
}
```

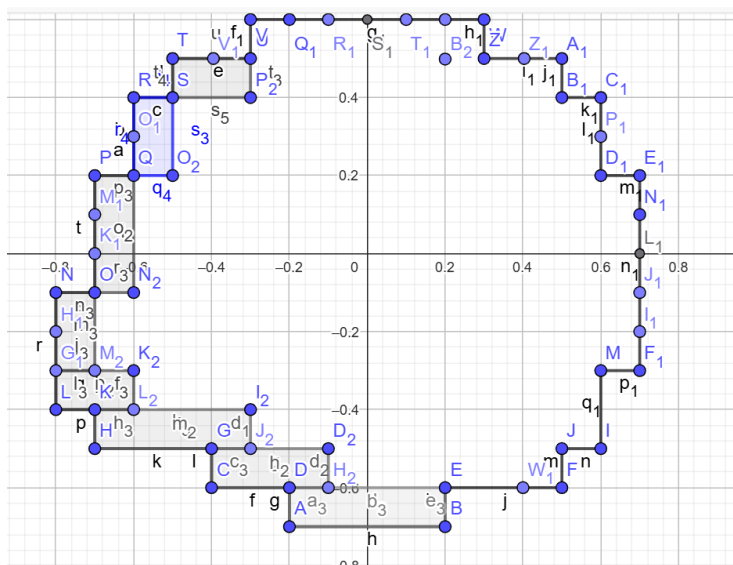
Para dibujar los vértices, dentro de la función main se hizo un ciclo for en donde cada pasada le indica a la función `glDrawArrays` el índice a partir del cual va a dibujar el polígono y el número de vértices a tomar. Como todos los vértices de la figura se van a encontrar en el arreglo “vertices” esta es la única función que se ocupará para dibujarlos.

```
float i = 0;
for (i = 0; i < sizeof(vertices); i = i + 4)
{
    glDrawArrays(GL_TRIANGLE_STRIP, i, 4);
}
```

Y de esta forma se construye el cuerpo del Boo



Para la parte de la sombra se hace el proceso anterior, construimos los polígonos en GeoGebra

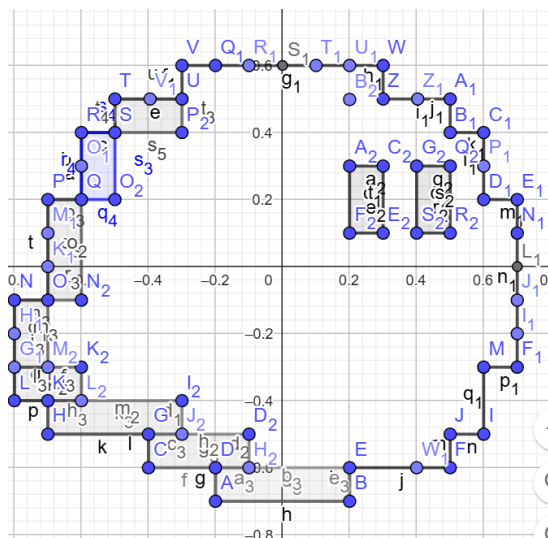


```
//sombras

//rectangulo 13
-0.2f, -0.7f, 0.0f, 0.5f,0.5f,0.5f,
-0.2f, -0.6f, 0.0f, 0.5f,0.5f,0.5f,
0.2f, -0.7f, 0.0f, 0.5f,0.5f,0.5f,
0.2f, -0.6f, 0.0f, 0.5f,0.5f,0.5f,

//rectangulo 14
-0.4f, -0.6f, 0.0f, 0.5f,0.5f,0.5f,
-0.4f, -0.5f, 0.0f, 0.5f,0.5f,0.5f,
-0.1f, -0.6f, 0.0f, 0.5f,0.5f,0.5f,
-0.1f, -0.5f, 0.0f, 0.5f,0.5f,0.5f,
```

A screenshot of a window titled "Practica 2. Leonardo Chagoya". The window contains a black background with a large, pixelated white shape in the center. The shape is composed of many small squares, giving it a blocky, digital appearance. The window has a standard title bar with a green icon on the left and minus, maximize, and close buttons on the right.



Vértices

```
// <<<<<OJOS>>>>>
//rectangulo 21
0.2f, 0.1f, 0.0f, 0.0f, 0.0f, 0.0f,
0.2f, 0.3f, 0.0f, 0.0f, 0.0f, 0.0f,
0.3f, 0.1f, 0.0f, 0.0f, 0.0f, 0.0f,
0.3f, 0.3f, 0.0f, 0.0f, 0.0f, 0.0f,

//rectangulo 22
0.4f, 0.1f, 0.0f, 0.0f, 0.0f, 0.0f,
0.4f, 0.3f, 0.0f, 0.0f, 0.0f, 0.0f,
0.5f, 0.1f, 0.0f, 0.0f, 0.0f, 0.0f,
0.5f, 0.3f, 0.0f, 0.0f, 0.0f, 0.0f,
```

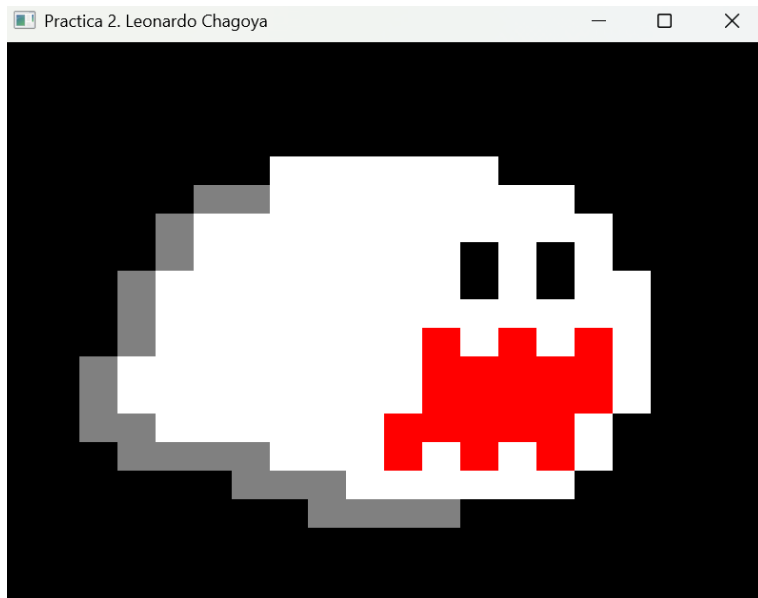
Resultado



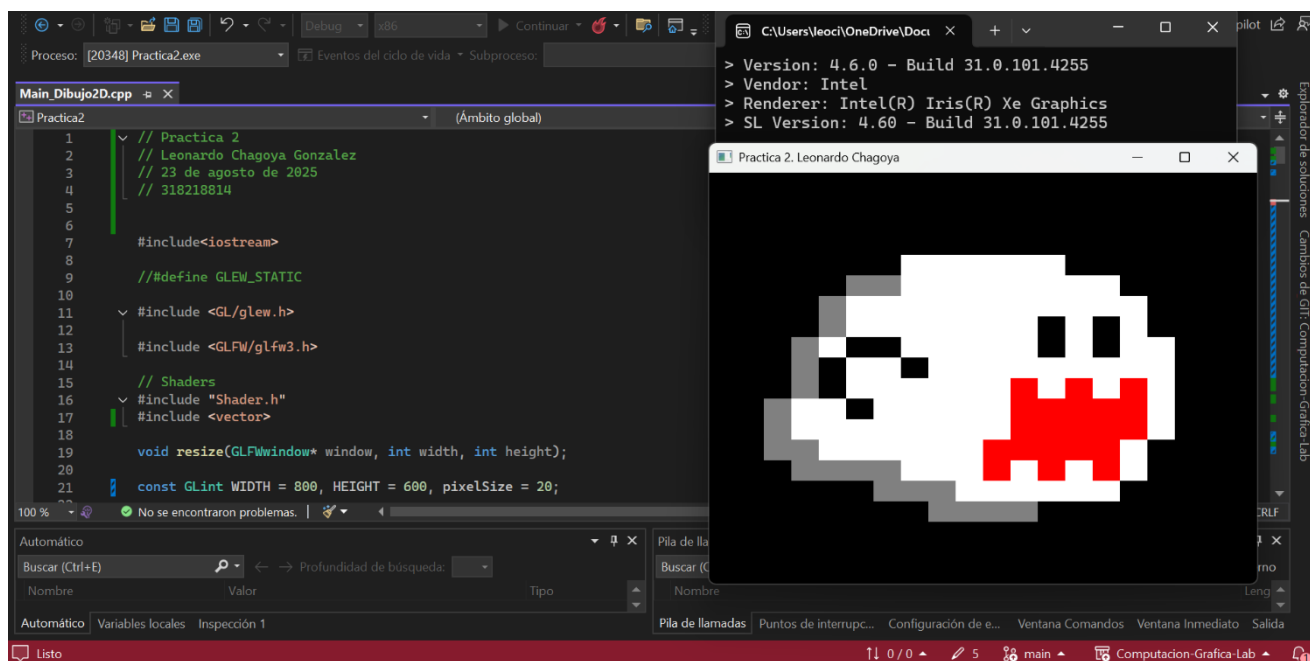
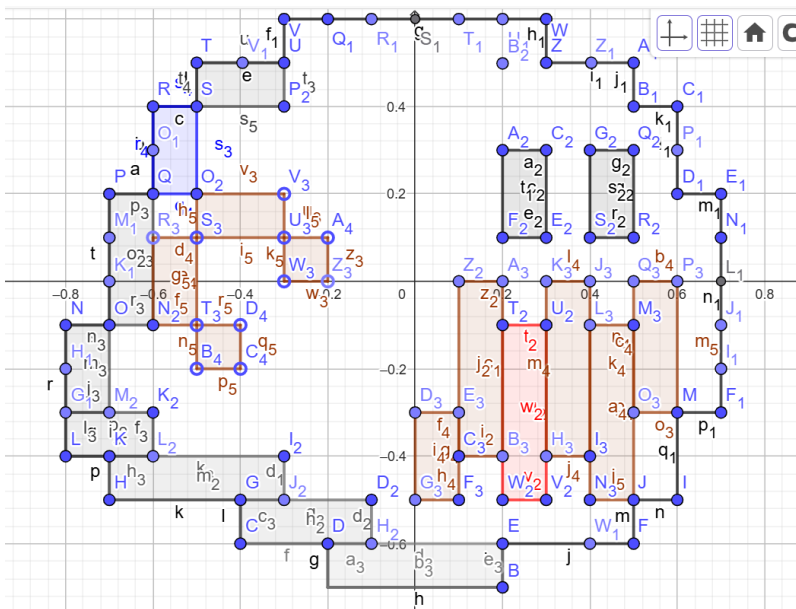
The diagram shows a 2D lattice with several distinct regions and labeled points. The horizontal axis is labeled with values -0.8, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8. The vertical axis is labeled with values -0.8, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8. The regions are defined by colored outlines and labels:

- Blue regions:** A large blue region on the left, a small blue square at the top center, and a blue square at the bottom center.
- Red regions:** A large red region on the right, a small red square at the top center, and a red square at the bottom center.
- Green regions:** A green square at the top center, a green square at the bottom center, and a green square at the bottom right.
- Yellow regions:** A yellow square at the top center, a yellow square at the bottom center, and a yellow square at the bottom right.
- Other regions:** A small grey square at the top center, a small grey square at the bottom center, and a small grey square at the bottom right.

Points are labeled with letters and subscripts, including A₁, A₂, A₃, A₄, B₁, B₂, B₃, B₄, C₁, C₂, C₃, C₄, D₁, D₂, D₃, D₄, E₁, E₂, E₃, E₄, F₁, F₂, F₃, F₄, G₁, G₂, G₃, G₄, H₁, H₂, H₃, H₄, I₁, I₂, I₃, I₄, J₁, J₂, J₃, J₄, K₁, K₂, K₃, K₄, L₁, L₂, L₃, L₄, M₁, M₂, M₃, M₄, N₁, N₂, N₃, N₄, O₁, O₂, O₃, O₄, P₁, P₂, P₃, P₄, Q₁, Q₂, Q₃, Q₄, R₁, R₂, R₃, R₄, S₁, S₂, S₃, S₄, T₁, T₂, T₃, T₄, U₁, U₂, U₃, U₄, V₁, V₂, V₃, V₄, W₁, W₂, W₃, W₄, X₁, X₂, X₃, X₄, Y₁, Y₂, Y₃, Y₄, Z₁, Z₂, Z₃, Z₄.



Construcción del detalle de la mano y resultado final



Conclusión/Reflexión

La práctica me gusto mucho, la opción de elegir con que imagen vas a trabajar me parece muy bien y la recomendación de buscar una figura que este con arte pixelArt ayuda al entendimiento de esta. La única dificultad que tuve es que al inicio quería hacerlo punto por punto y después me di cuenta que era más fácil hacerlo con polígonos.

Repositorio de GitHub

<https://github.com/chagoya27/Computacion-Grafica-Lab>

Bibliografía

OpenGL. (2020, 19 de Julio). Primitive. <https://www.khronos.org/opengl/wiki/primitive>